УНИВЕРСИТЕТ ИТМО

Д.В. Кайсарова, И.Ю. Коцюба

МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА.

Санкт-Петербург
2016

Кайсарова Д.В., Коцюба И.Ю. Математическая лингвистика. Практикум. – СПб: Университет ИТМО, 2016. – 67 с.

В практикуме представлены материалы практических работ по курсу «Математическая лингвистика».

Практикум разработан для студентов направления подготовки 45.03.04 – «Сетевые интеллектуальные системы в гуманитарной сфере».

Рекомендовано к печати решением совета факультета инфокоммуникационных технологий

(Протокол № 2 от 25 февраля 2016 года).

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Оглавление

Введение

Практические занятия по дисциплине «Математическая лингвистика» разработаны на основе существующих стандартов, рекомендаций разработчиков и ведущих специалистов в области применения программных средств, используемых в лабораторных практикумах.

Математические модели и информационные технологии прочно вошли в область лингвистических исследований. Трудно найти область лингвистики, где бы ни решались задачи автоматической обработки. В настоящее время арсенал математической лингвистики включает в себя методы корпусной лингвистики, автоматического перевода текстов, автореферирования, вопросно-ответных систем, распознавания и синтеза речи, обработки естественного языка, создания онтологий и электронных словарей, автоматического извлечения фактов из текста и многие другие. Наличие высококвалифицированных специалистов в области математической лингвистики позволяет организациям решать самые сложные задачи по анализу лингвистических данных.

Методическое пособие включает шесть практических работ и итоговый семинар. Весь практикум рассчитан на применение активных технологий обучения как в группах, так и самостоятельной работы студентов. Практикум по дисциплине «Математическая лингвистика» разработан для студентов направления подготовки 45.03.04 – «Интеллектуальные системы в гуманитарной сфере», но будет полезен студентам других направлений подготовки, которые интересуются методами автоматизированной обработки лингвистической информации.

Практическая работа № 1

«Анализ лингвистических явлений с точки зрения возможности их математического моделирования»

Цель практической работы: Проанализировать данное математическое явление и указать сложности, которые могут возникнуть при его математическом моделировании.

Задачи практической работы:

1. Ознакомиться с выбранным лингвистическим явлением, решив одну или несколько лингвистических задач;
2. Разработать методический материал, поясняющий суть явления, на основе решенных задач;
3. Добавить примеры данного явления из русского, английского, других языков;
4. Описать сложности, которые могут возникнуть при математическом моделировании данного явления;
5. Выступить с докладом на семинаре; руководить работой группы по анализу данного явления и сложностей его математического моделирования.

Справочный материал: Задачи лингвистических олимпиад 1965 – 1975 гг. М., изд-во МНЦМО, 2007. Для анализа предлагаются следующие задачи: 5, 12, 19, 26, 38, 40, 41, 45, 46, 56, 57, 62, 74, 75, 77, 81, 82, 87, 91, 98, 99, 100, 106, 117, 121, 126, 132, 134, 136, 139, 143, 145, 158, 159, 171, 174, 183, 185, 195, 237, 198, 200, 201.

Образец выполнения задания:

Задача № 120. Условие:

Даны формы азербайджанского глагола с переводом на русский язык

Задание 1: опишите порядок следования морфем.

Задание 2: переведите следующие формы глагола на азербайджанский язык.

Решение:

Слайд 1

1) **бахмаг** — смотреть (инфинитив)
2) **бахабилмамаг** — **не мочь** смотреть (инфинитив)
3) **бахыраммы** — смотрю **ли я**? (**наст. вр.**)
4) **бахышабилырлар** — **они могут** смотреть **друг на друга** (**наст. вр.**)
5) **бахмадылар** — **они не** смотрели (**прош. вр.**)

6) **бахдырабилдымы** — **мог ли** он **заставлять** смотреть? (**прош. вр.**)
7) **бахмалыдысан** — **ты должен** был смотреть (**прош. вр.**)
8) **бахдырырам** — **я заставляю** смотреть (**наст. вр.**)
9) **бахмасады** — **если** он **не** смотрел (**прош. вр.**)
Смотришь **ли ты**?
Они не смотре**ли друг на друга**.
Заставлять смотреть.
Если он **мог** смотреть. (**прош. вр.**)
Слайд 2
1) **бахмаг** — смотреть (инфинитив)
2) **бахабилмамаг** — **не мочь** смотреть (инфинитив)
3) **бахырамммы** — смотрю **ли я**? (**наст. вр.**)
4) **бахышабилырлар** — **они могут** смотреть **друг на друга** (**наст. вр.**)
5) **бахмадылар** — **они не** смотрели (**прош. вр.**)
6) **бахдырабилдымы** — **мог ли** он **заставлять** смотреть? (**прош. вр.**)
7) **бахмалыдысан** — **ты должен** был смотреть (**прош. вр.**)
8) **бахдырырам** — **я заставляю** смотреть (**наст. вр.**)
9) **бахмасады** — **если** он **не** смотрел (**прош. вр.**)

1	2	3	4	5	6	7	8	9
Бах	(ыш, малы)	абил	ма	са	(ыр, ды)	(мы, сан, лар)	мы	маг

Табл. 1. Упорядочивание морфем

Слайд 3
1) **бахмаг** — смотреть (инфинитив)
2) **бахабилмамаг** — **не мочь** смотреть (инфинитив)
3) **бахырамммы** — смотрю **ли я**? (**наст. вр.**)
4) **бахышабилырлар** — **они могут** смотреть **друг на друга** (**наст. вр.**)
5) **бахмадылар** — **они не** смотрели (**прош. вр.**)
6) **бахдырабилдымы** — **мог ли** он **заставлять** смотреть? (**прош. вр.**)
7) **бах*малы*дысан** — **ты *должен*** был смотреть (**прош. вр.**)
8) **бахдырырам** — **я заставляю** смотреть (**наст. вр.**)
9) **бахмасады** — **если** он **не** смотрел (**прош. вр.**)

Смотришь **ли ты**? — бахсанмы
Они не смотре**ли друг на друга**. — **бахышмадылар**
Заставлять смотреть. — **бахдырмаг**
Если он **мог** смотреть. (**прош. вр.**) — **бахабилсады**

Обсуждение

Эта задача иллюстрирует следующие трудности математического моделирования естественного языка:

1. Значения, выраженные в азербайджанском языке грамматически (формой слова), в русском языке выражены лексически (отдельным словом).
    a) При переводе с русского языка на азербайджанский следует выделить группу русских слов, которая будет соответствовать одному русскому слову;
    b) Следует определять грамматические категории у всей группы в целом, а не у отдельных слов (группа «мог ли он заставлять смотреть?» содержит два формальных показателя инфинитива, однако эта группа как целое имеет третье лицо и единственное число, и, таким образом, не стоит в инфинитиве);
    c) При переводе на русский язык следует определить корректный порядок слов, который не будет соответствовать порядку морфем азербайджанского слова.

2. Следует учитывать порядок следования морфем в слове.
    a) Правила расстановки касаются не только отдельных морфем, но и групп морфем (показатели лица и числа – предпоследнее место; показатели модальности и взаимности – второе место);
    b) Наличие одних морфем исключает появление других (показатель инфинитива исключает наличие показателей лица и числа).

3. В русском языке в формах третьего лица единственного числа прошедшего времени род выражен грамматически, то есть говорящий обязан указать, какого рода действующее лицо;
    a) при переводе с азербайджанского языка следует принимать во внимание род подлежащего или использовать местоимение типа «некто» или безличные конструкции

4. Некоторые грамматические значения выражены нулевой морфемой: **бахмасады — если** <u>он</u> **не** смотрел.
    a) При переводе с азербайджанского языка придется сформулировать правило типа «если присутствует показатель времени и отсутствует показатель лица, при переводе следует использовать местоимение 3 лица ед. ч».

5. Экспоненты некоторых морфем частично совпадают:
    a) ма – отрицание, **малы** – долженствование;
    b) ыр – настоящее время, дыр – заставлять.

Пример 1

<u>Задача №145</u>

Перевести на эстонский язык:

1. Он не положил котла в реку, а я не положил хлеба в комнату.

2. Он построил сени и сани.

Исходные данные

1. Ma joonistasin mäe, jõe, oru ja raja, aga ta ei joonistanud mäge, jõge, orgu ega rada.
Я нарисовал гору, реку, долину и тропу, а он не нарисовал ни горы, ни реки, ни долины, ни тропы.
2. Ta avastas selle tõve, aga ma ei avastanud seda tõbe.
Он обнаружил эту болезнь, а я не обнаружил этой болезни.
3. Ta sattus tõppe.
Он впал в болезнь.
4. Ma asetasin paja kotta, siis ta asetas leiva patta.
Я положил котёл в сени, тогда он положил хлеб в котёл.
5. Ma ei ehitanud tuba, koda ega rege.
Я не построил ни комнаты, ни сеней, ни саней.

Анализ фраз

1. **Ma** joonistasin mäe, jõe, oru ja raja, aga **ta** ei joonistanud mäge, jõge, orgu ega rada.
Я нарисовал гору, реку, долину и тропу, а **он** не нарисовал ни горы, ни реки, ни долины, ни тропы.
2. **Ta** avastas selle tõve, aga **ma** ei avastanud seda tõbe.
Он обнаружил эту болезнь, а **я** не обнаружил этой болезни.
3. **Ta** sattus tõppe.
Он впал в болезнь.
4. **Ma** asetasin paja kotta, siis **ta** asetas leiva patta.
Я положил котёл в сени, тогда **он** положил хлеб в котёл.
5. **Ma** ei ehitanud tuba, koda ega rege.
Я не построил ни комнаты, ни сеней, ни саней.

Анализ морфем

1. **Ma** *joonista*sin mäe, jõe, oru ja raja, aga **ta** ei *joonista*nud mäge, jõge, orgu ega rada.
Я *нарисовал* гору, реку, долину и тропу, а **он** не *нарисовал* ни горы, ни реки, ни долины, ни тропы.
2. **Ta** *avasta*s selle tõve, aga **ma** ei *avasta*nud seda tõbe.
Он *обнаружил* эту болезнь, а **я** не *обнаружил* этой болезни.
3. **Ta** *satt*us tõppe.
Он *впал* в болезнь.
4. **Ma** *aseta*sin paja kotta, siis **ta** *aseta*s leiva patta.
Я *положил* котёл в сени, тогда **он** *положил* хлеб в котёл.
5. **Ma** ei *ehita*nud tuba, koda ega rege.
Я не *построил* ни комнаты, ни сеней, ни саней.

Винительный (что?)	**Родительный (нет чего?)**	**Винительный (во что?)**	**Перевод**
mäe	**mäge**	mäkke	гора
jõe	**jõge**	jõkke	река
oru	**orge**	orkke	долина
raja	ra**d**a	ra**tt**a	тропа
tõve	*tõ**b**e*	*tõ**pp**e*	болезнь
paja	pa**d**a	pa**tt**a	котёл
leiva	*lei**b**a*	*lei**pp**a*	хлеб
tuva	*tu**b**a*	*tu**pp**a*	комната
koja	ko**d**a	ko**tt**a	сени
ree	**rege**	rekke	сани

Табл. 2. Сравнительная таблица существительных

1. **Он** не положил *котла* в реку, а **я** не положил *хлеба* в комнату.
Та ei asetanud *pada* jõkke, aga **ma** ei asetanud *leiba* tuppa.
2. **Он** построил *сени* и *сани*.
Та ehitas *koja* ja *ree*.

Комментарий проблемы

1. Эстонский имеет большее количество падежей, чем русский, и следует учитывать контекст, в котором используется существительное, а не только его форму.
2. При этом в эстонском не используются предлоги, указывающие на падеж, изменяется форма самого слова.
3. Глаголы прошедшего времени зависят не только от числа, но и от лица подлежащего. При этом отрицательная форма имеет и частицу, и суффикс.
4. Для указания на отсутствие объекта, в русском языке используется частица «ни»; эстонский имеет специальный падеж для этого.
5. Для понимания того, как образуется один из падежей, требуется знание фонетики, что исключает использование только визуального анализа текста.

Практическая работа № 2

«Построение алгоритмов анализа лингвистических явлений»

Цель практической работы: Закрепление теоретических знаний в вопросах моделирования лингвистических явлений.

Задачи практической работы: Построить пошаговый алгоритм анализа данного лингвистического явления, подобрать справочные данные, необходимые для его реализации; указать места возможных ошибок, объяснить их природу, указать способы минимизации вероятности их появления.

Задание для практической работы

1. Ознакомиться с описанием алгоритма анализа лингвистического явления (явлений).
2. Описать:
    a. Цель данного вида анализа;
    b. Область применения;
    c. Основные определения.
3. Привести краткое словесное описание алгоритма;
4. Создать блок-схему алгоритма;
5. Подобрать необходимые для выполнения алгоритма пакеты данных;
6. Проверить алгоритм на предоставленных преподавателем данных;
7. Обнаружить места вероятного появления ошибок, описать характеристики исходных данных, при которых появление ошибок наиболее вероятно;
8. Предложить методы устранения или минимизации ошибок, описать их ограничения;
9. Проверить алгоритм на самостоятельно выбранных исходных данных, объяснить, почему выбраны именно эти данные;
10. Подготовиться к выступлению на семинаре с сообщением о процессе и результате практической работы.

Пример 1

Задание: создать блок-схему алгоритма сравнения двух списков Сводеша для оценки степени родства языков.

Цель: оценить время разделения двух данных языков, или время прекращения существования их общего предка.

Область применения: уточнение степени родства языков, если данное родство в принципе допускается экспертным сообществом (конвенциональное родство).

Определения

Языковое родство — происхождение языков от одного общего языка-предка. Языки, являющиеся результатами различных путей эволюции одного праязыка, называются **родственными** и характеризуются регулярными соответствиями на различных уровнях, объяснимых общностью происхождения, а не случайным совпадением или заимствованием: их исконные морфемы находятся в строго определённых соответствиях, отражающих действие исторических звуковых изменений.

Список Сводеша — предложенный американским лингвистом Моррисом Сводешем инструмент для оценки степени родства между различными языками по такому признаку, как схожесть наиболее устойчивого базового словаря. Представляет собой стандартизированный перечень базовых лексем данного языка, приблизительно упорядоченный по убыванию их исторической устойчивости. Минимальный набор важнейшей («стержневой») лексики содержится в 100-словном списке Сводеша. Используется также более устаревший и менее семантически устойчивый, но зато более подробный 200-словный список.

Описание алгоритма:

1. На первом этапе выясняется звучание всех слов из обоих списков, то есть составляются их транскрипции;
2. Далее проводится предварительное сопоставление транскрипций, выясняются регулярные соответствия между звуками двух данных языков.
3. Слова сравниваются попарно. Подсчитывается количество транскрипций совпадающих с точностью до регулярных соответствий.
4. Исходя из количества совпадений высчитывается время распада общего предка (14 слов из 100-словного списка за тысячелетие согласно).

Необходимые данные: Правила составления транскрипции для языков из выбранной пары.

Данные для проверки (набор, предоставленный преподавателем): список Сводеша русского языка, список Сводеша английского языка.

Результаты проверки: 5 совпадений, предполагаемое время распада общего предка – V тысячелетие до нашей эры.

Причины возникновения ошибок:

1. Алгоритм не учитывает синонимические замены
2. При небольшом количестве совпадений не удается выявить надежные регулярные звуковые соответствия

Предложения по устранению ошибок:

1. Обращать внимание не только на звуковую, но и на смысловую сторону сравниваемых слов.
2. При небольшом количестве совпадений использовать не 100-словный, а 200-словный список.

Данные для проверки (самостоятельно выбранный набор): список Сводеша русского языка, список Сводеша польского языка. Распад общего предка русского и польского языков надежно засвидетельствован источниками, поэтому выводы сравнения списков Сводеша можно будет проверить по внешним сведениям и таким образом оценить надежность алгоритма.

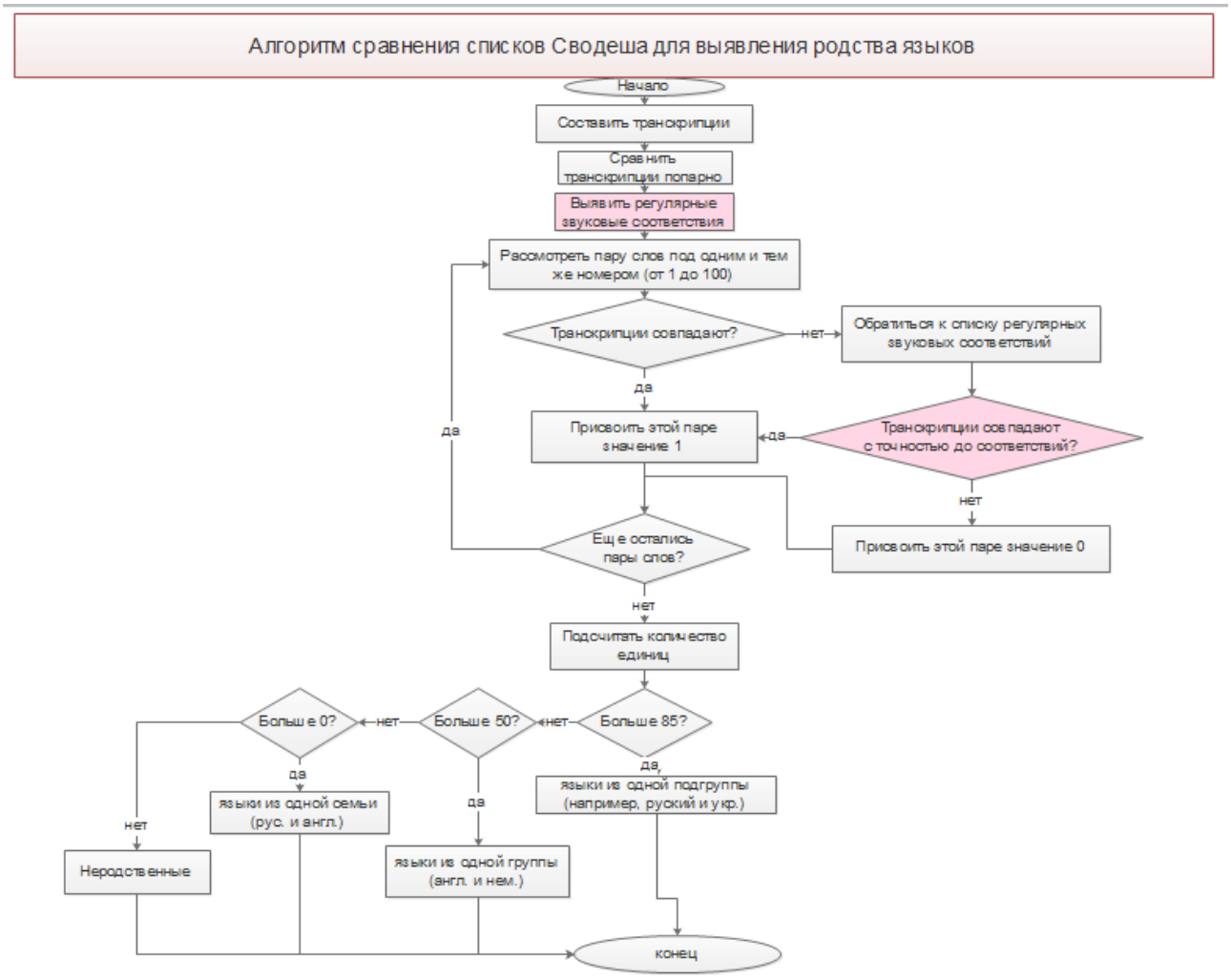


Рис. 1. Блок-схема алгоритма примера 1

Пример 2

Задание: создать блок-схему алгоритма работы программы, генерирующей кальки к словам, заимствованным из латинского и древнегреческого языков.

Область применения: изучение значений морфем, заимствованных из латинского и древнегреческого языков в рамках изучения русского языка; изучение чередований на стыке морфем.

Определения:
Калька (от фр. *Calque* – копия) в лингвистике – заимствование иностранных слов, выражений, фраз непосредственным переводом определенной языковой единицы, в том числе, результат таких заимствований: слова, словосочетания и фразы.

Причины возникновения ошибок:

1. Некоторые группы звуков могут принадлежать как приставкам, так и корням;
2. Наличие интерфикса в структуре некоторых слов.

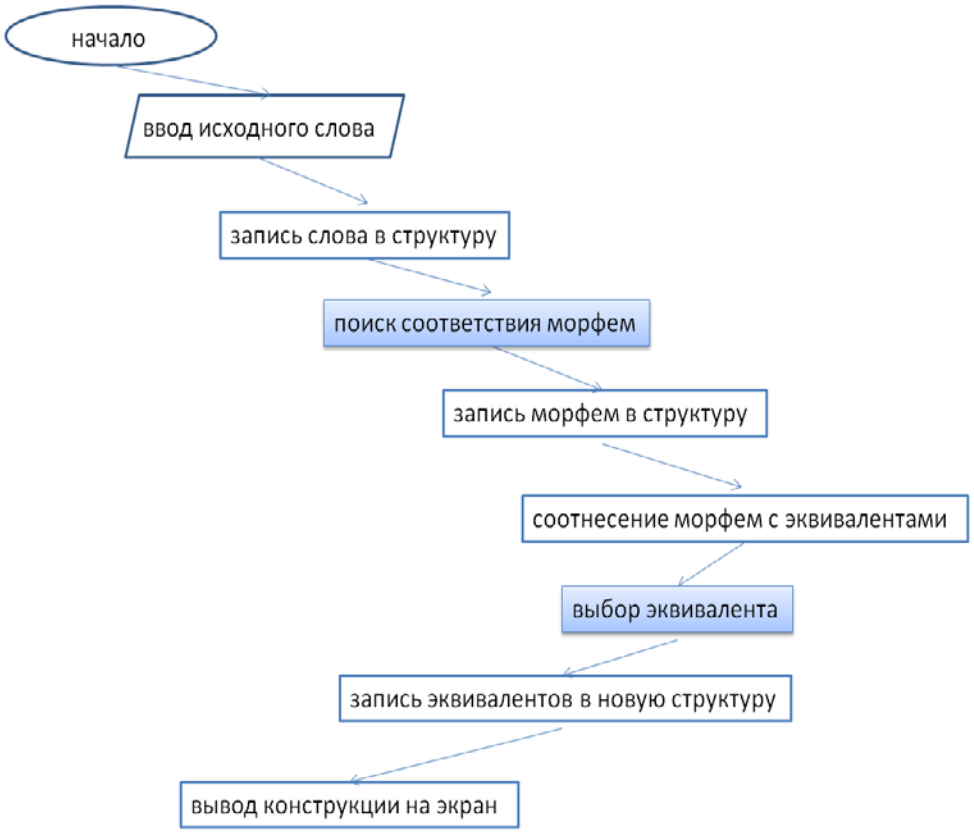
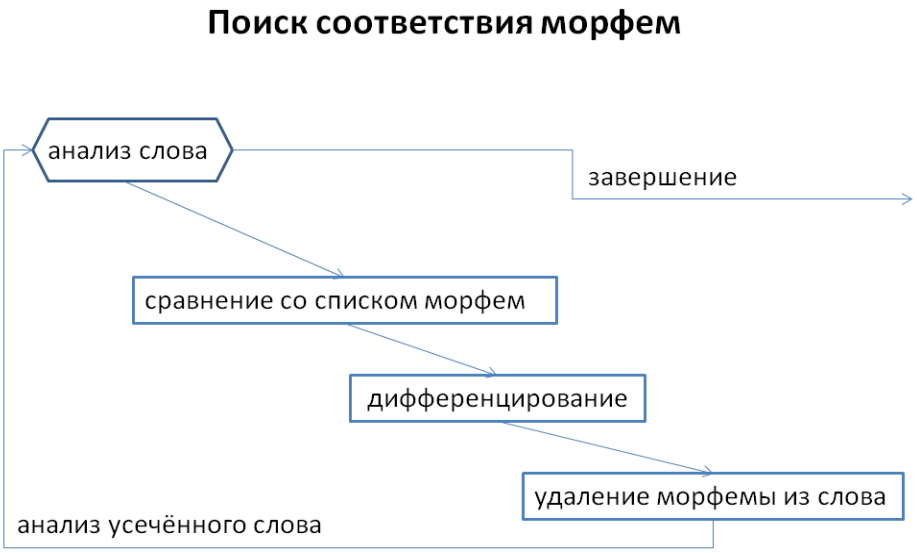


Рис. 2. Блок-схема алгоритма примера 2



Примечание: поиск осуществляется побуквенно с начала слова

Рис. 3. Поиск соответствия морфем

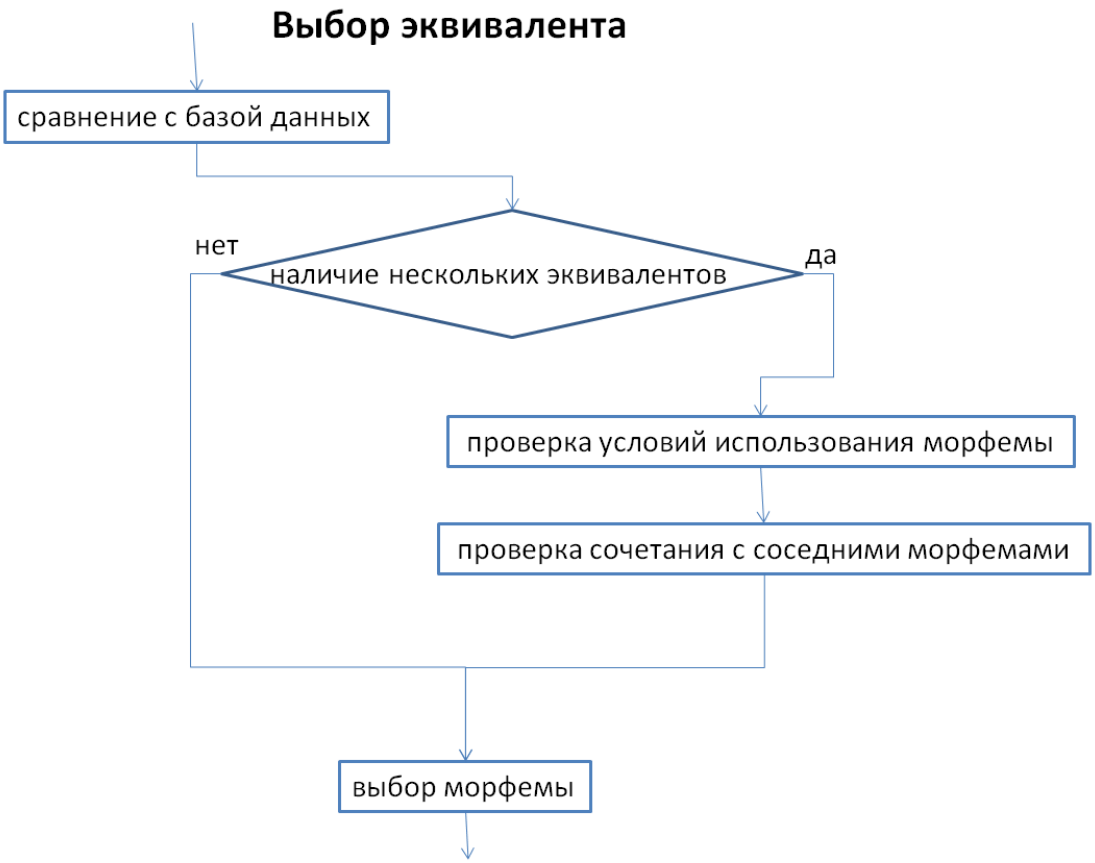


Рис. 4. Выбор эквивалента

Практическая работа № 3

«Анализ программного кода с использованием лексических методов»

Цель практической работы: Закрепление теоретических знаний в вопросах анализа программного кода с использованием лексических методов как направления математической лингвистики.

Задание для практической работы: Ознакомиться с заданием на практическую работу. Выполнить учебное задание, сделать выводы.

Задание для практической работы

1. Разработать алгоритмы расчета метрик лексического анализа программного кода с использованием метрик Холстеда, Чепина и т.д.
2. Проанализировать полученные результаты, оценить точность полученных значений, сделать выводы.

Пример 1. Построить алгоритм, отражающий последовательность действий по расчету количества непустых и пустых строк программного кода, количества строк с комментариями и их процентное отношение к общему числу строк кода. Пример представлен на рис. 5-8.

Пример 2. Построить алгоритм, отражающий последовательность действий по расчету метрик Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

1. P - вводимые переменные для расчетов и для обеспечения вывода.

Примером может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы, т.е. сама переменная не модифицируется, а только содержит исходную информацию.

2. M - модифицируемые, или создаваемые внутри программы переменные.
3. C - переменные, участвующие в управлении работой программного модуля (управляющие переменные).
4. T - не используемые в программе («паразитные») переменные. Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Далее вводится значение метрики Чепина :

Q = a1*P + a2*M + a3*C + a4*T,
где a1, a2, a3, a4 - весовые коэффициенты. Примеры алгоритмов на рис. 9.

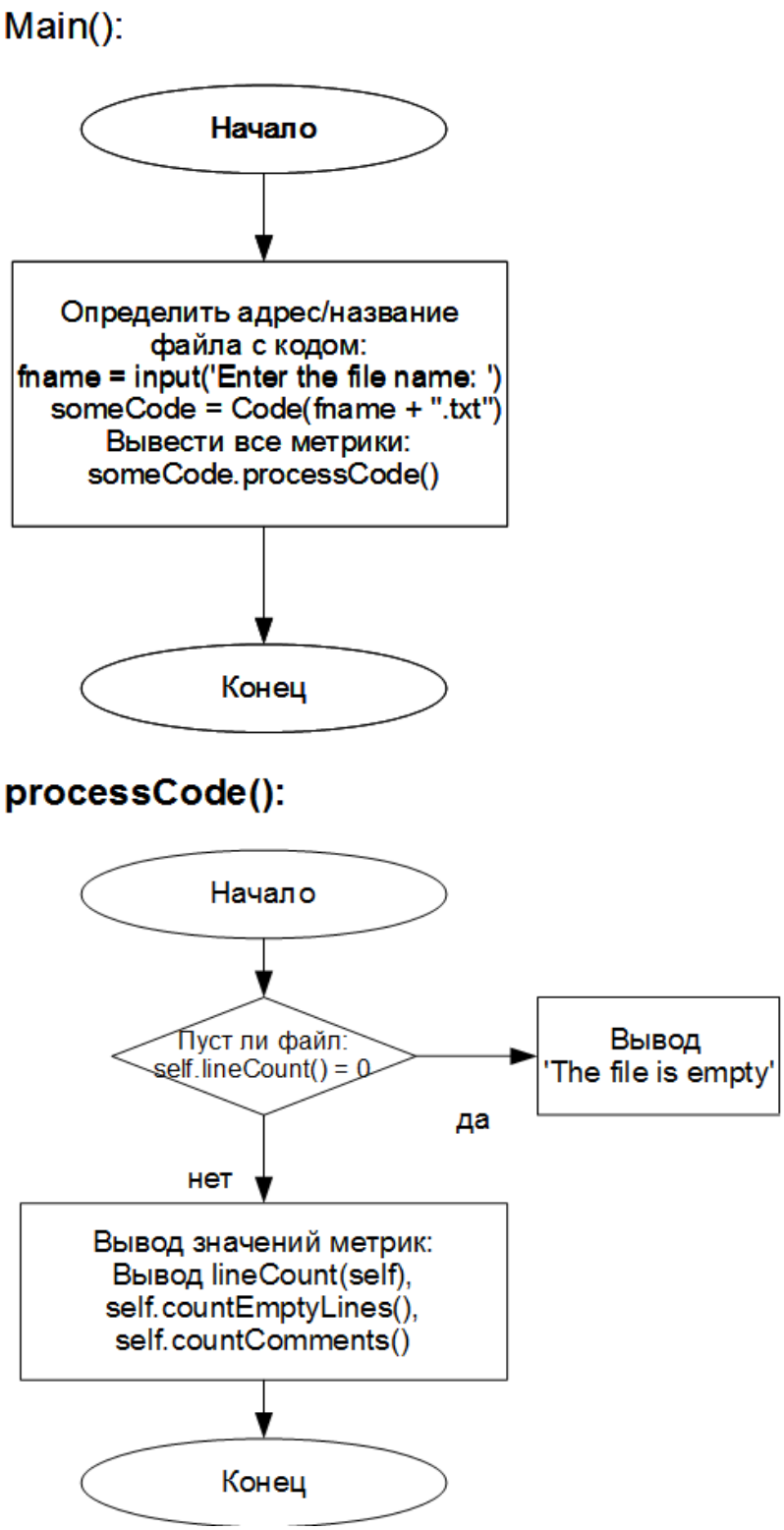


Рис. 5. Алгоритм вывода метрик

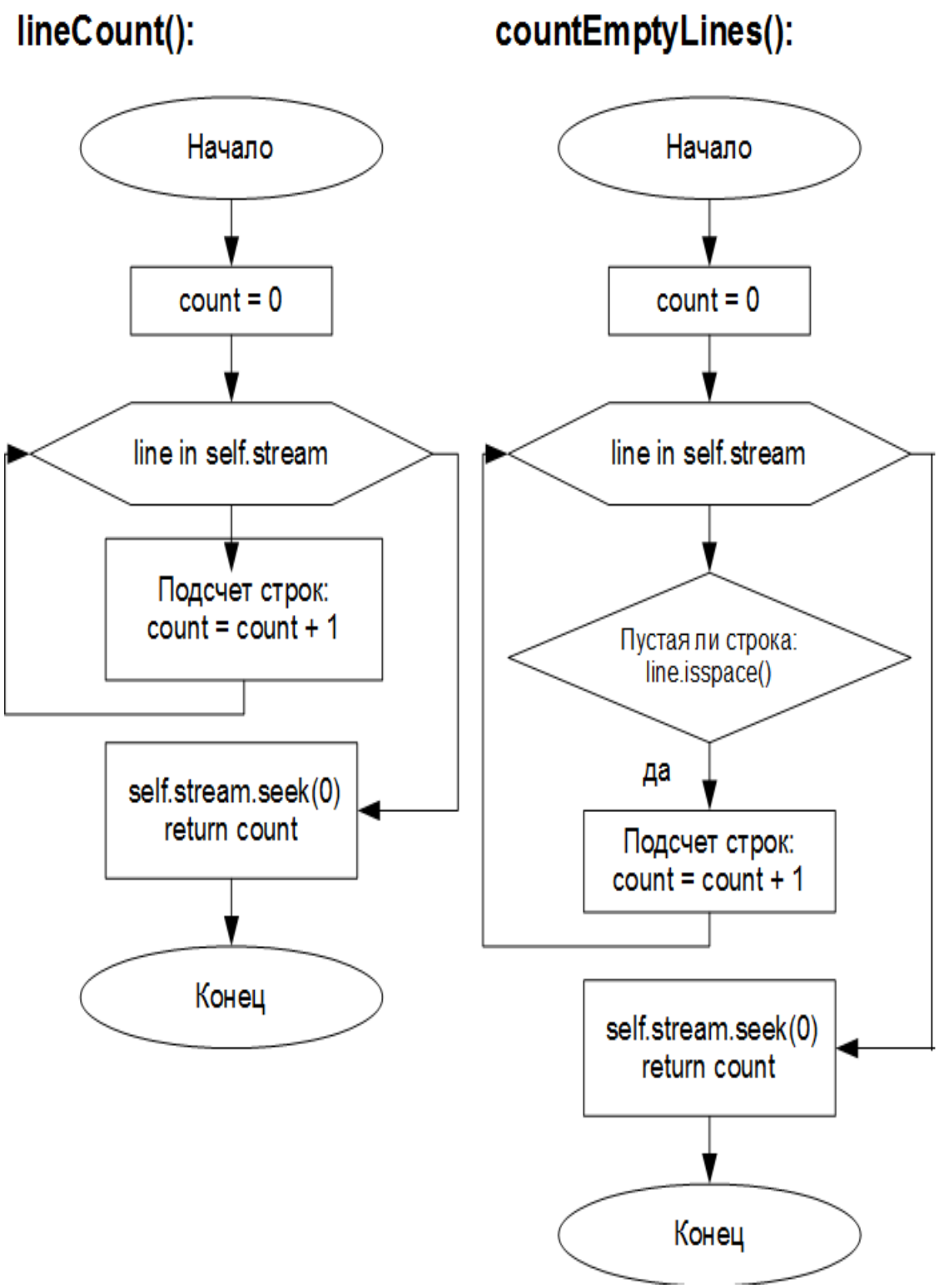


Рис. 6. Алгоритм подсчета строк

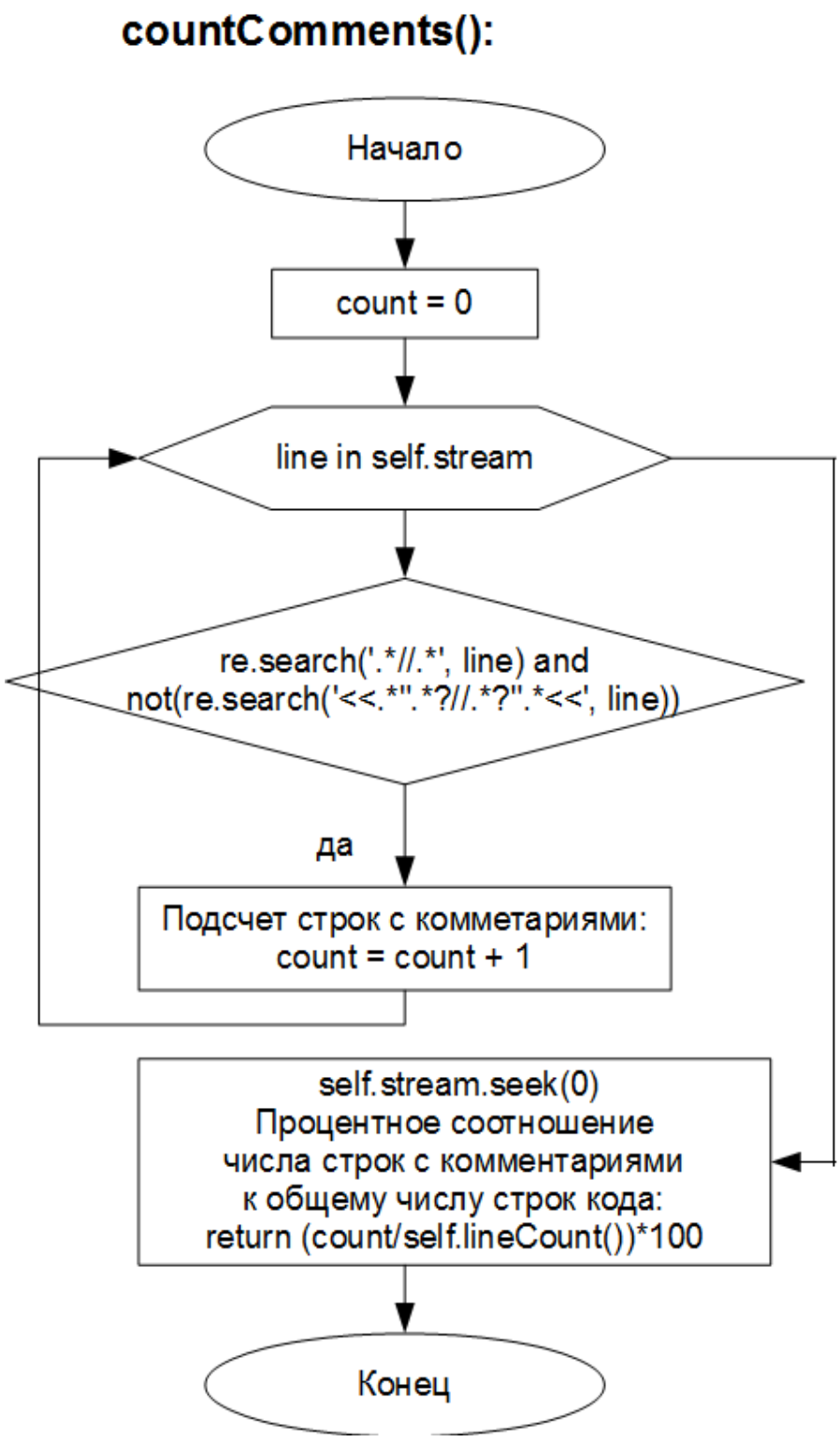


Рис. 7. Алгоритм подсчета строк с комментариями

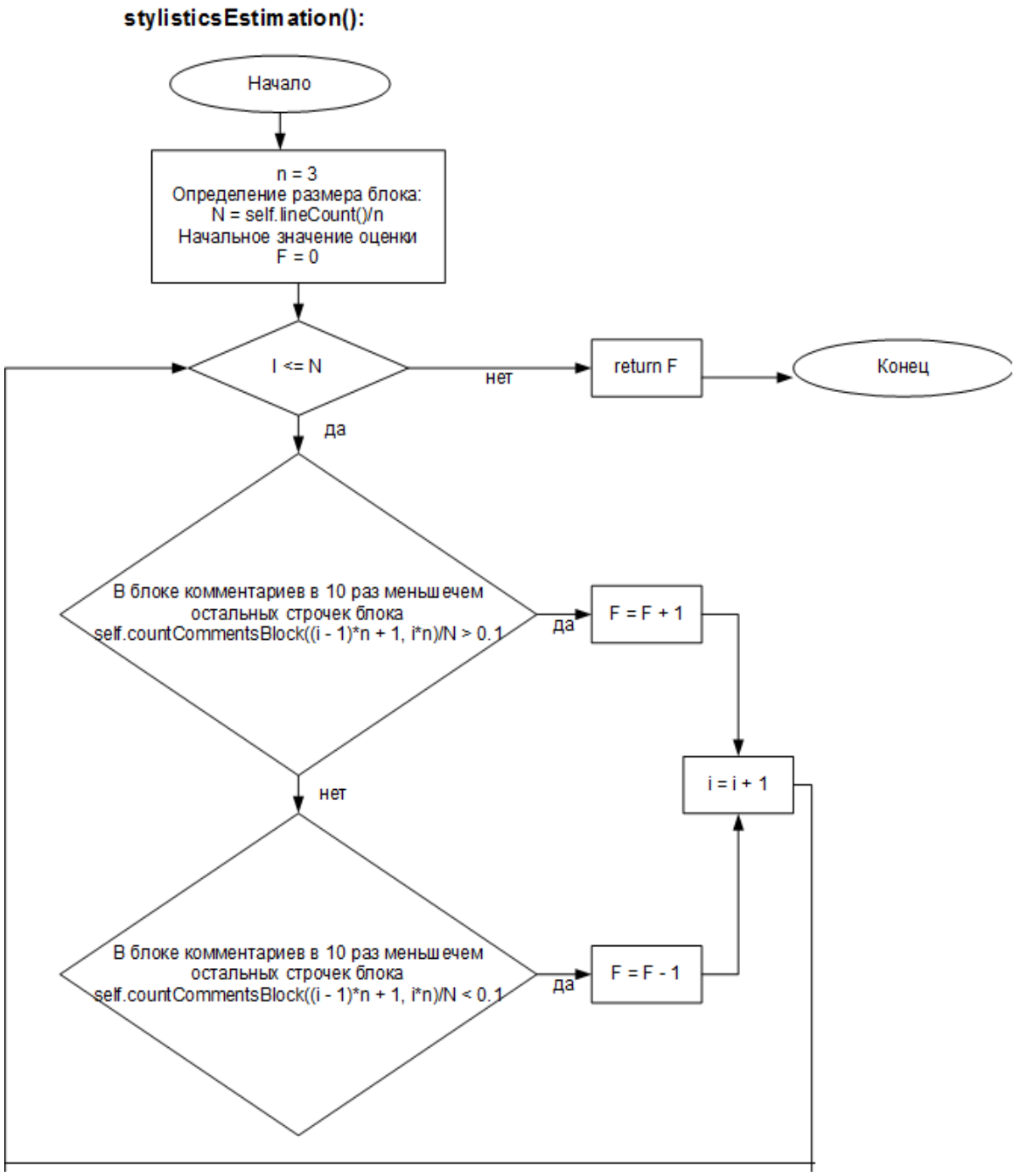


Рис. 8. Алгоритм подсчета количества комментарием

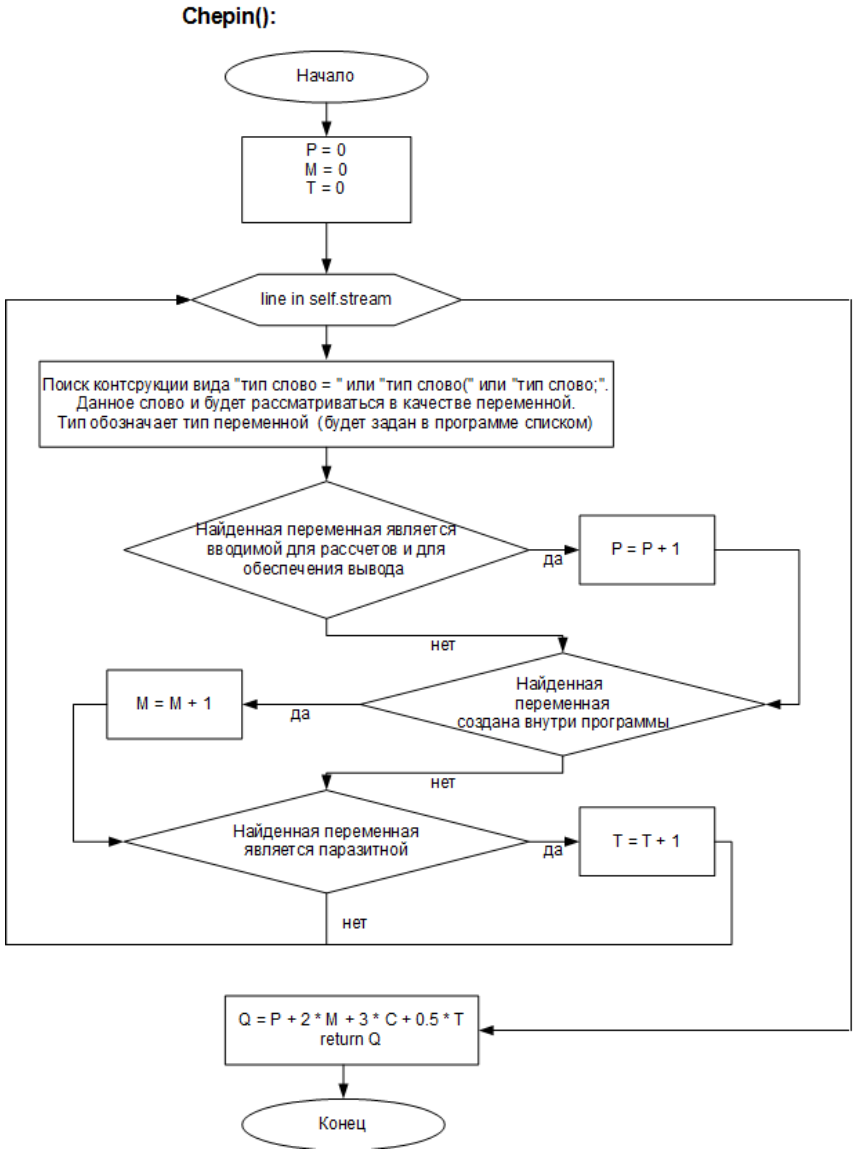


Рис. 9. Алгоритм расчета метрик Чепина

Пример 3. Построить алгоритм, отражающий последовательность действий по расчету метрик Холстеда с использованием следующих обозначений:
n=n1+n2 — словарь программы,

N=N1+N2 — длина программы,

n'=n1'+n2' — теоретический словарь программы,

N'= n1*log2(n1) + n2*log2(n2) — теоретическая длина программы (для стилистически

корректных программ отклонение N от N' не превышает 10)

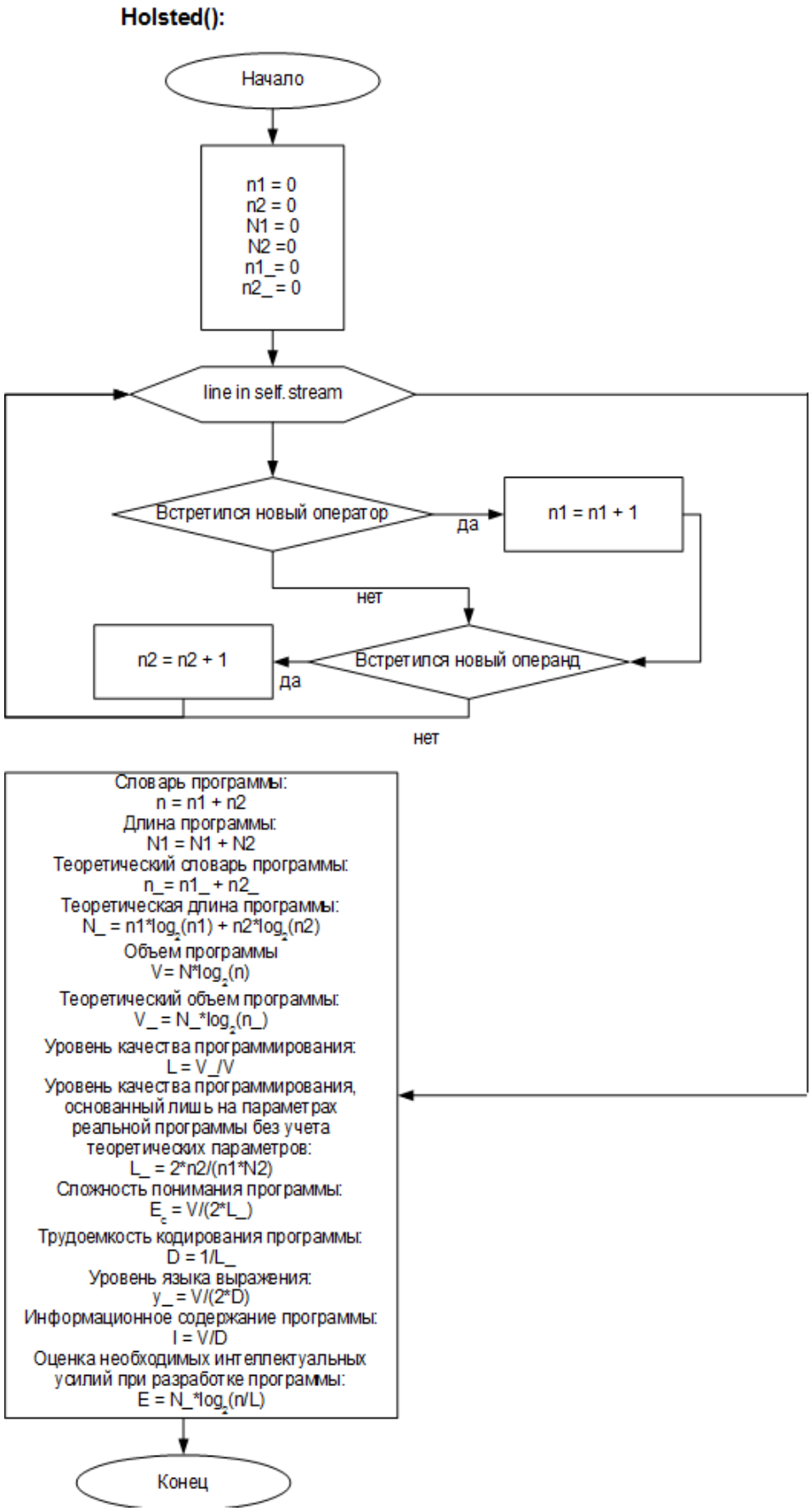


Рис. 10. Алгоритм расчета метрик Холстеда

Практическая работа № 4

«Статистическая обработка текстов»

Цель практической работы: Закрепление теоретических знаний в вопросах статистической обработки текстов как направления квантитативной лингвистики.

Задание для практической работы: Ознакомиться с репрезентативными наборами лингвистических данных, выбрать данные для статистического анализа, определить гипотезу статистического анализа. Выполнить учебное задание по анализу данных, сделать выводы.

Задание для практической работы

1. На основе выбранных лингвистических данных определить гипотезу исследования, изучить выбранный набор данных на репрезентативность, при необходимости проанализировать данные на предмет пропусков, выполнить предобработку данных.
2. Проанализировать полученные результаты, оценить точность полученных значений, сделать выводы.

Пример 1. В качестве выборки был взят набор стихотворений А.А.Ахматовой. Необходимо по представленному набору определить размер и такие тропы, как эпитеты, метафоры, олицетворения и сравнения, проанализировать их соотношение.

Набор данных:

«Сжала руки под тёмной вуалью…»

Сжала руки под тёмной вуалью…
«Отчего ты сегодня бледна?»
- Оттого, что я терпкой печалью
Напоила его допьяна.

Как забуду? Он вышел, шатаясь,
Искривился мучительно рот…
Я сбежала, перил не касаясь,
Я бежала за ним до ворот.

Задыхаясь, я крикнула: «Шутка
Всё, что было. Уйдешь, я умру.»
Улыбнулся спокойно и жутко
И сказал мне: «Не стой на ветру».

Мужество

Мы знаем, что ныне лежит на весах
И что совершается ныне.
Час мужества пробил на наших часах,
И **мужество нас не покинет.**
Не страшно под пулями мертвыми лечь,
Не горько остаться без крова,
И мы сохраним тебя, русская речь,
Великое русское слово.
Свободным и чистым тебя **пронесем,**
И внукам **дадим**, и от плена **спасем**
Навеки.

Сероглазый король

Слава тебе, **безысходная боль**!
Умер вчера сероглазый король.

Вечер **осенний был душен и ал,**
Муж мой, вернувшись, спокойно сказал:

«Знаешь, с охоты его принесли,
Тело у старого дуба нашли.

Жаль королеву. Такой молодой!..
За ночь одну она стала седой».

Трубку свою на камине нашел
И на работу ночную ушел.

Дочку мою я сейчас разбужу,
В серые глазки ее погляжу.

А за окном **шелестят тополя:**
«Нет на земле твоего короля…»

Сад

Он весь **сверкает и хрустит,**
Обледенелый сад.
Ушедший от меня грустит,
Но нет пути назад.
И **солнца бледный тусклый лик** —
Лишь круглое **окно;**

Я тайно знаю, чей двойник
Приник к нему давно.
Здесь мой **покой навеки взят**
Предчувствием беды,
Сквозь тонкий лед еще **сквозят**
Вчерашние следы.
Склонился тусклый мертвый лик
К **немому сну полей,**
И **замирает острый крик**
Отсталых журавлей.

Белой ночью

Ах, дверь не запирала я,
Не зажигала свеч,
Не знаешь, как, усталая,
Я не решалась лечь.
Смотреть, **как гаснут полосы**
В закатном мраке хвой,
Пьянея звуком голоса,
Похожего на твой.
И знать, что все потеряно,
Что жизнь - **проклятый ад!**
О, я была уверена,
Что ты придёшь назад.

Вечером

Звенела музыка в саду
Таким **невыразимым горем.**
Свежо и остро пахли морем
На блюде устрицы во льду.
Он мне сказал: «Я **верный друг**!» —
И моего коснулся платья.
Как не похожи на объятья
Прикосновенья этих рук.
Так гладят кошек или птиц,
Так на наездниц смотрят стройных...
Лишь **смех в глазах** его **спокойных**
Под легким **золотом ресниц.**
А **скорбных скрипок голоса**
Поют за стелющимся дымом:
«Благослови же небеса —
Ты первый раз одна с любимым».

Рыбак

Руки голы выше локтя,
А глаза синей**, чем лед.**
Едкий, душный запах дегтя,
Как загар, тебе идет.
И всегда, всегда распахнут
Ворот куртки голубой,
И рыбачки только ахнут,
Закрасневшись пред тобой.
Даже девочка, что ходит
В город продавать камсу,
Как потерянная бродит
Вечерами на мысу.
Щеки бледны, руки слабы,
Истомленный взор глубок,
Ноги ей **щекочут крабы,**
Выползая на песок.
Но она уже не ловит
Их протянутой рукой.
Все сильней **биенье крови**
В теле **раненном тоской.**

Песня последней встречи

Так **беспомощно грудь холодела**,
Но шаги мои **были легки.**
Я на правую руку надела
Перчатку с левой руки.
Показалось, что много ступеней,
А я знала — их только три!
Между кленов **шепот осенний**
Попросил: «Со мною умри!
Я **обманут** моей **унылой,**
Переменчивой, злой судьбой».
Я ответила: «Милый, милый!
И я тоже. Умру с тобой...»
Эта **песня** последней встречи.
Я взглянула на темный дом.
Только в спальне горели свечи
Равнодушно-желтым огнем.

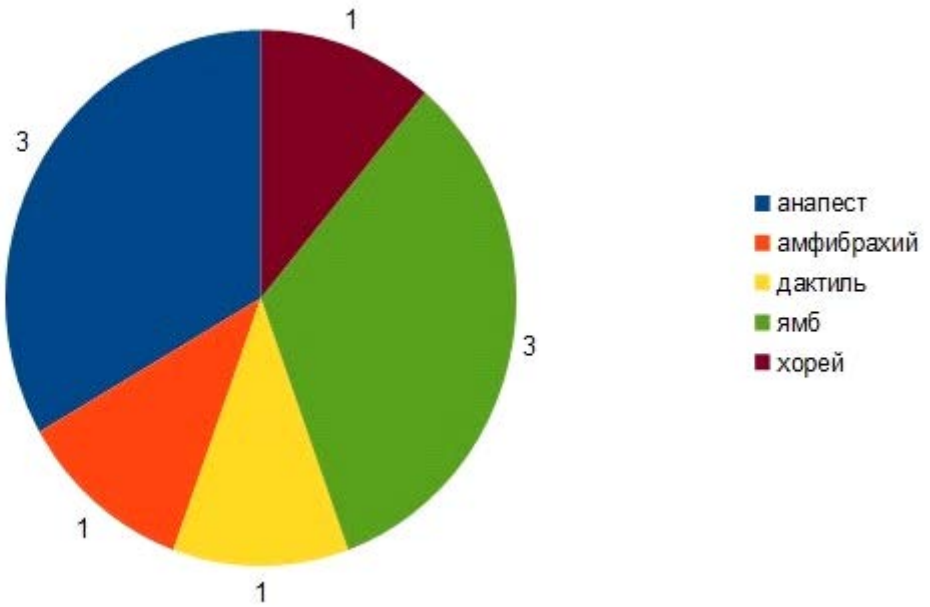
Двадцать первое. Ночь. Понедельник.

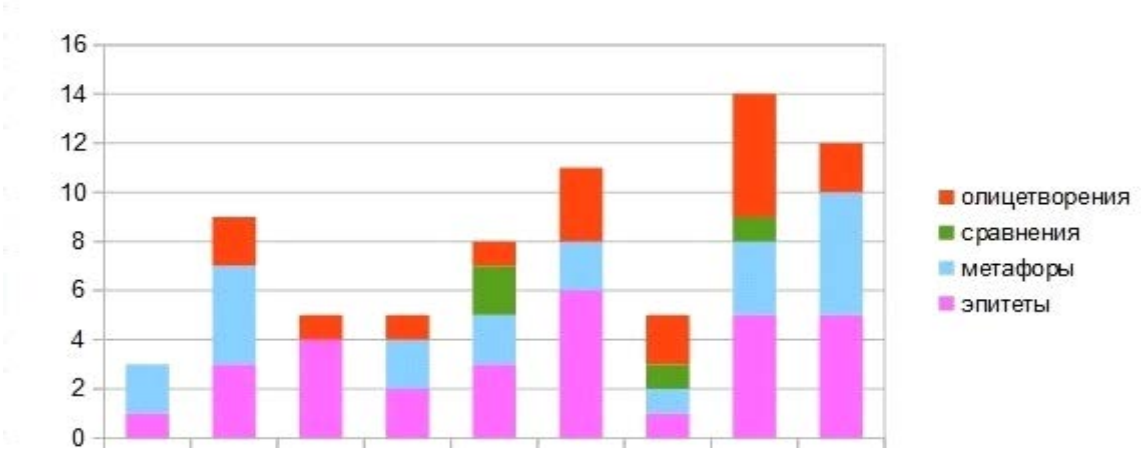
Двадцать первое. Ночь. Понедельник.
Очертанья столицы во мгле.
Сочинил же какой-то бездельник,
Что бывает любовь на земле.
И от лености или со скуки
Все поверили, так и живут:
Ждут свиданий, боятся разлуки
И **любовные песни** поют.
Но иным **открывается тайна,**
И **почиет** на них тишина...Я на это наткнулась случайно
И с тех пор все **как будто больна.**

Далее все данные были сведены в таблицу с указанием названия стихотворения, его размера, соотношения троп:

стихотворение	размер	эпитеты	метафоры	сравнения	олицетворения
Сжала руки под темной вуалью	анапест	1	2	0	0
Мужество	амфибрахий	3	4	0	2
Сероглазый король	дактиль	4	0	0	1
Белой ночью	ямб	2	2	0	1
Рыбак	хорей	3	2	2	1
Песня последней встречи	анапест	6	2	0	3
Двадцать первое. Ночь. Понедельник	анапест	1	1	1	2
Сад	ямб	5	3	1	5
Вечером	ямб	5	5	0	2

Соотношение размеров стихов и соотношение литературных троп:





Вывод:

По построенным диаграммам заметно, что в стихах больше эпитетов, чем других средств выразительности. Как и метафоры (которых замечено тоже немало), они ёмкие и как никто другой обогащают текст. Преимущество описаний (эпитетов) можно объяснить тем, что большая часть стихотворений – любовная лирика, которая полна описанием чувств, переживаний лирической героини. На большем наборе данных может быть более точно сделан вывод о тенденциях употребления размера стихов поэтессы.

Пример 2. Для анализа были выбраны 2 стихотворения, часто анализируемые в среде лингвистов: «Памятник» А.С.Пушкина и «Памятник» Г.Р.Державина. По представленным стихотворениям необходимо провести частотный анализ и определить сходство-различие стихотворений по лексическому составу; определить, какие части речи преобладают в стихотворении каждого поэта и одинаковы ли эти преобладания.

Тексты стихотворений:

Пушкин:

Я памятник себе воздвиг нерукотворный,
К нему не зарастет народная тропа,
Вознесся выше он главою непокорной
Александрийского столпа.
Нет, весь я не умру — душа в заветной лире
Мой прах переживет и тленья убежит —
И славен буду я, доколь в подлунном мире
Жив будет хоть один пиит.
Слух обо мне пройдет по всей Руси великой,
И назовет меня всяк сущий в ней язык,
И гордый внук славян, и финн, и ныне дикой
Тунгус, и друг степей калмык.

И долго буду тем любезен я народу,
Что чувства добрые я лирой пробуждал,
Что в мой жестокий век восславил я Свободу
И милость к падшим призывал.
Веленью божию, о муза, будь послушна,
Обиды не страшась, не требуя венца,
Хвалу и клевету приемли равнодушно
И не оспоривай глупца.

Державин:
Я памятник себе воздвиг чудесный, вечный,
Металлов тверже он и выше пирамид;
Ни вихрь его, ни гром не сломит быстротечный,
И времени полет его не сокрушит.
Так! — весь я не умру, но часть меня большая,
От тлена убежав, по смерти станет жить,
И слава возрастет моя, не увядая,
Доколь славянов род вселенна будет чтить.
Слух пройдет обо мне от Белых вод до Черных,
Где Волга, Дон, Нева, с Рифея льет Урал;
Всяк будет помнить то в народах неисчетных,
Как из безвестности я тем известен стал,
Что первый я дерзнул в забавном русском слоге
О добродетелях Фелицы возгласить,
В сердечной простоте беседовать о боге
И истину царям с улыбкой говорить.
О муза! возгордись заслугой справедливой,
И презрит кто тебя, сама тех презирай;
Непринужденною рукой неторопливой
Чело твое зарей бессмертия венчай.

Проведем частотный анализ данных. Фрагмент таблицы представлен ниже:

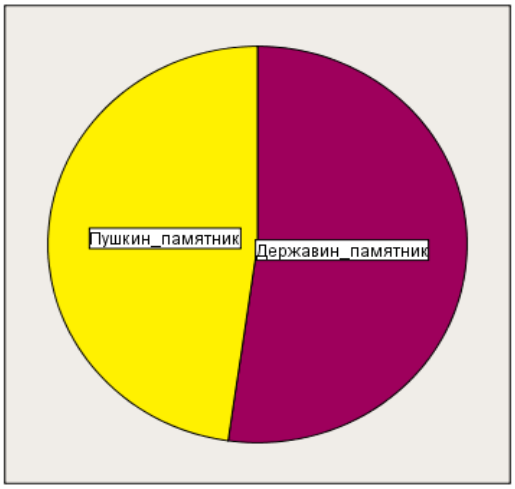
Пушкин:

	Frequency	Percent	Valid Percent	Cumulative Percent
Александрийско	1	,8	,8	5,4
божию	1	,8	,8	6,2
будет	1	,8	,8	7,0
буду	2	1,6	1,6	8,5
будь	1	,8	,8	9,3
в	4	3,1	3,1	12,4
век	1	,8	,8	13,2
веленью	1	,8	,8	14,0
великой	1	,8	,8	14,7
венца	1	,8	,8	15,5
весь	1	,8	,8	16,3
внук	1	,8	,8	17,1
воздвиг	1	,8	,8	17,8
вознесся	1	,8	,8	18,6
восславил	1	,8	,8	19,4
всей	1	,8	,8	20,2
всяк	1	,8	,8	20,9
выше	1	,8	,8	21,7
главою	1	,8	,8	22,5
глупца	1	,8	,8	23,3
гордый	1	,8	,8	24,0
Total	123	100,0	100,0	

Державин:

	Frequency	Percent	Valid Percent	Cumulative Percent
царям	1	,8	,8	,8
безвестности	1	,8	,8	1,6
Белых	1	,8	,8	2,3
беседовать	1	,8	,8	3,1
бессмертия	1	,8	,8	3,9
боге	1	,8	,8	4,7
большая	1	,8	,8	5,4
будет	2	1,6	1,6	7,0
быстроте	1	,8	,8	7,8
в	3	2,3	2,3	10,1
венчай	1	,8	,8	10,9
весь	1	,8	,8	11,6
вечный	1	,8	,8	12,4
вихрь	1	,8	,8	13,2
вод	1	,8	,8	14,0
возгласить	1	,8	,8	14,7
возгордись	1	,8	,8	15,5
воздвиг	1	,8	,8	16,3
возрастет	1	,8	,8	17,1
Волга	1	,8	,8	17,8
времени	1	,8	,8	18,6
вселенна	1	,8	,8	19,4
Total	129	100,0	100,0	

Путем группировки слов по частям речи и анализа их соотношения получаем диаграмму:

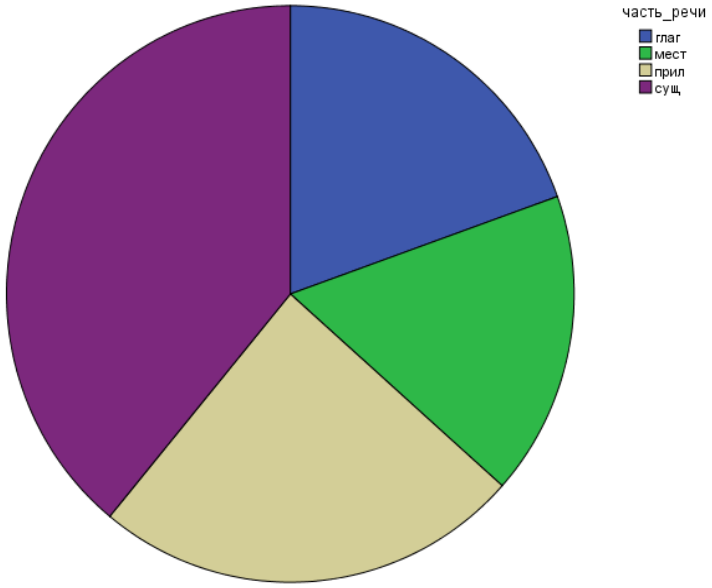


Проанализируем соотношение общих слов в данных стихотворениях:

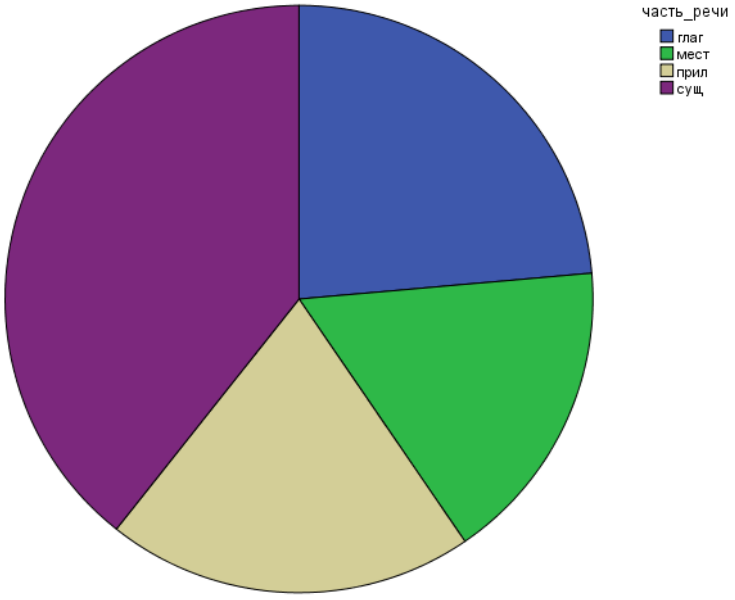
Слово	Частота_Пушкин	Частота_Державин
муза	1	1
народ	1	1
памятник	1	1
славяне	1	1
слух	1	1
весь	1	1
всякий	1	1
высокий	1	1
быть	4	2
воздвигнуть	1	1
пройти	1	1
убежать	1	1
умереть	1	1
я	6	4
себе	1	1
он	1	1
мне	1	1
меня	1	1
мой	2	1
я	6	4
доколь	1	1

Проиллюстрируем соотношение основных частей речи в каждом стихотворении:

«Памятник» А.С.Пушкин



«Памятник» Г.Р. Державин



Вывод:

В результате частотного анализа можно сделать вывод о том, что, несмотря на смысловую схожесть, общую тему и одинаковое количество строф и строк в

каждой строфе стихотворений А.С Пушкина и Г.Р. Державина, они сильно отличаются по лексическому составу. Это подтверждается малым количеством слов, одинаковых для двух произведений - их всего 20, что составляет около 0,1% от общего числа.

К тому же, на основании построенных диаграмм нетрудно заметить, что в произведениях по-разному соотносятся части речи: у Пушкина преобладают существительные и прилагательные, а у Державина - существительные и глаголы.

Пример 3. На небольшой выборке проверить гипотезу о том, что у Л.Н. Толстого в произведении «Война и мир» употреблены предложения, которые на порядок длиннее и сложнее предложений других писателей. Возьмем по 10 предложений из трех произведений: «Война и мир» Толстой Л.Н., «Обломов» Гончаров И.А., «Леди Макбет Мценского уезда» Лесков Н.С. и построим их синтаксические деревья. В табличных данных будет представлены информация о количестве листьев дерева, уровней и составных предложений.

Толстой

№ предложения	1	2	3	4	5	6	7	8	9	10
Кол-во слов	57	49	71	58	74	49	79	74	51	83
Кол-во уровней	11	9	10	10	9	9	6	8	10	8
Кол-во подпредложений	5	5	2	4	4	3	10	7	4	9

Лесков

№ предложения	1	2	3	4	5	6	7	8	9	10
Кол-во слов	35	52	35	89	68	50	49	40	52	41
Кол-во уровней	9	10	8	8	8	7	7	9	10	12
Кол-во подпредложений	4	3	5	6	8	6	6	3	7	3

Гончаров

№ предложения	1	2	3	4	5	6	7	8	9	10
Кол-во слов	92	42	67	91	46	73	37	67	42	61
Кол-во уровней	9	9	7	11	10	10	7	12	10	10
Кол-во подпредложений	15	3	4	6	4	4	2	4	4	5

Общее

	Толстой	Лесков	Гончаров
Кол-во слов мах	83	89	**92**
Кол-во слов сред	**65**	51	61
Кол-во подпредложений мах	10	8	**15**

Диаграмма количества слов в предложениях:

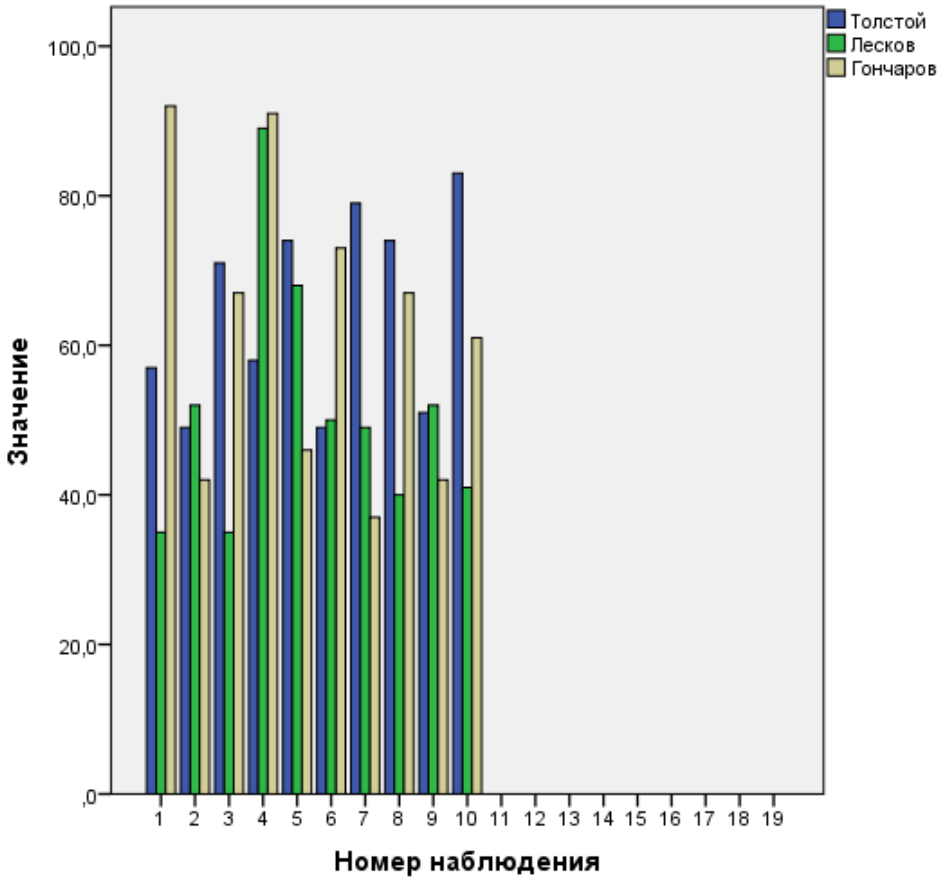


Диаграмма количества уровней в предложениях:

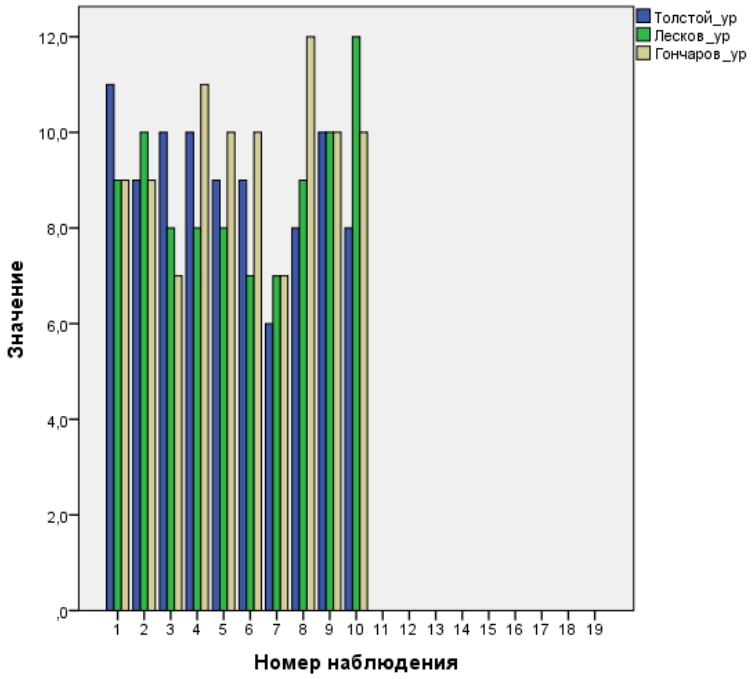
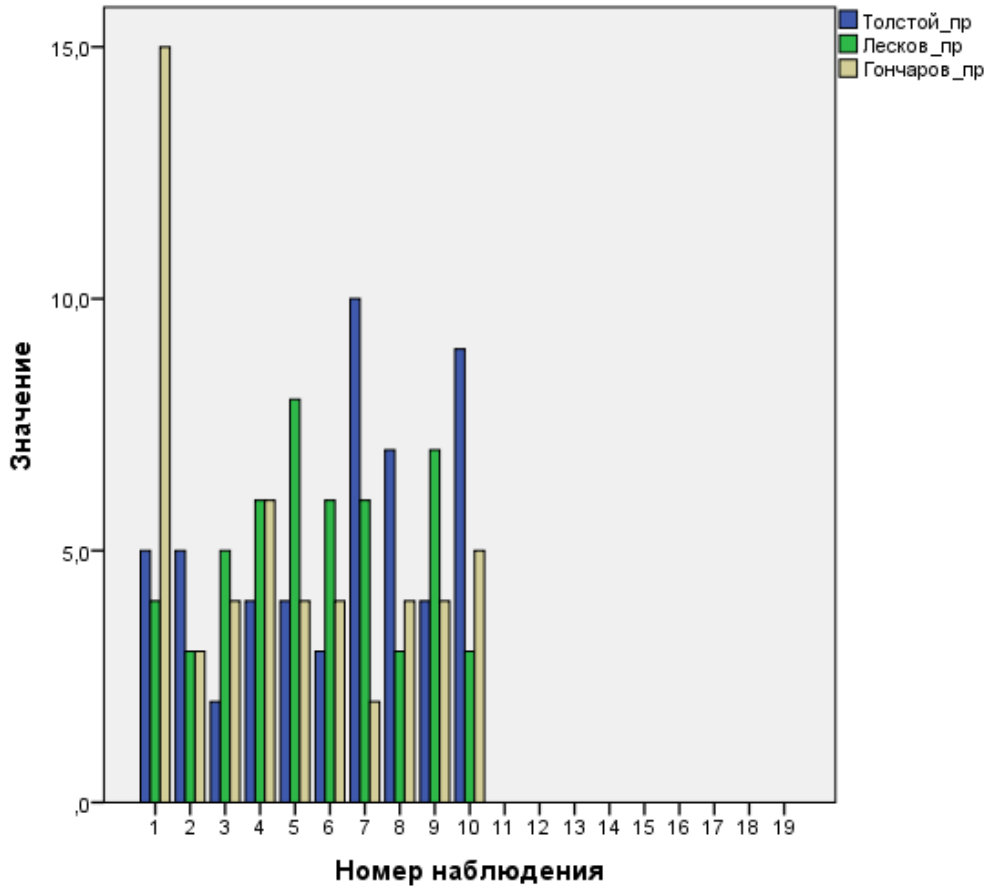


Диаграмма количества составных предложений:



Вывод:

Если судить по максимальным значениям, то самое длинное и сложное предложение оказалось у Гончарова. Если же смотреть по средним показателям, то выдвигаемая гипотеза, что у Толстого предложения длиннее и сложнее, не отвергается.

Практическая работа № 5

«Автоматический анализ лингвистической информации»

Цель практической работы: Закрепление теоретических знаний в вопросах автоматического анализа текстов и другой лингвистической информации как направления математической лингвистики.

Задачи практической работы: Ознакомиться с методами автоматического анализа лингвистической информации. Выполнить учебное задание по написанию программы для решения какого-либо вопроса математической лингвистики, сделать выводы.

Задание для практической работы

1. На основе выбранной задачи изучить технологии ее решения, подготовить данные для обработки.
2. Написать программу, реализующую решение выбранной задачи в автоматическом режиме.

Пример. Написать программу, реализующую очищение текста от служебных частей речи (союзы, предлоги, артикли). Предварительно составить корпус наиболее частотных из них или взять готовый корпус.

Листинг программы с комментариями представлен ниже:

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <cstdlib>
#include <ctype.h>
#include <string.h>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <cstring>      // для строк
#include <string>
#include <string.h>
#include <stdlib.h>
using namespace std;

int main()
{
        string a;
        fstream F;
        ofstream f;

        string artikels[] = {" a ", " an ", " the "};
```

```
        string predlogi[] = {" aboard ", " about ", " above ", " absent ", " across ", " afore ", " after ", "
against ", " along ", " amid ", " amidst ", " among ", " amongst ", " around ", " as ", " aside ", " aslant ",
" astride ", " at ", " athwart ", " atop ", " before ", " behind ", " below ", " beneath ", " beside ", "
besides ", " between ", " betwixt ", " by ", " circa ", " despite ", " down ", " except ", " from ", " in ", "
inside ", " into ", " mid ", " near ", " neath ", " next ", " of ", " off ", " on ", " opposite ", " outside
", " over ", " pace ", " per ", " round ", " to ", " toward ", " towards ", " under ", " underneath ", "
unlike ", " up ", " via ", " with ", " without ", " during ", " excepting ", " excluding ", " including ", "
alongside ", " within ", " upon ", " onto ", " throughout ", " wherewith ", " according to ", " ahead of ", "
apart from ", " as far as ", " as for ", " as of ", " as per ", " as regards ", " aside from ", " as well as
", " away from ", " because of ", " by force of ", " by means of ", " close to ", " contrary to ", " due to ",
" except for ", " far from ", " for the sake of ", " in accordance with ", " in addition to ", " in case of ",
" in connection with ", " in consequence of ", " in front of ", " in spite of ", " in the back of ", " in the
course of ", " in the event of ", " in the middle of ", " in to ", " into ", " inside of ", " instead of ", "
in view of ", " near to ", " next to ", " on account of ", " on to ", " onto ", " on top of ", " opposite to
", " out of ", " outside of ", " owing to ", " thanks to ", " up to ", " with regard to ", " with respect to
"};
        string soyuzi[] = {" and ", " as weel as ", " but ", " either ", " neither ", " or ", " while ", "
that ", " if ", " whelther ", " after ", " as ", " as long as ", " as soon as ", " before ", " since ", "
untill ", " till ", " when ", " while ", " as ", " because ", " for ", " now ", " on the ground that ", " that
", " in case ", " once ", " on condition that ", " so as long as ", " supposing that ", " unless ", " for fear
that ", " lest ", " so that ", " in order that ", " than ", " in spite of the fact that ", " notwithstanding
that ", " thought ", " althought ", " moreover ", " nevertheless ", " nor ", " on the contary ", " on the
other hand ", " otherwise ", " therefore ", " that is why ", " still ", " thus ", " while ", " yet ", " not
yet ", " for ", " with ", " without ", " within ", " according to ", " owing to ", " in spite of ", " in terms
of ", " on behalf of ", " meanwhile ", " from now on ", " from time to time ", " beyond ", " whereas ", " at
least ", " at last ", " on condition "};

        int art_siz = sizeof(artikels)/sizeof(artikels[0]);             // вычисляем длину массивов
        int pred_siz = sizeof(predlogi)/sizeof(predlogi[0]);
        int soy_siz = sizeof(soyuzi)/sizeof(soyuzi[0]);

        F.open("text.txt");                     //открываем файл в режиме чтения
        f.open("out.txt", ios::out);            // в режиме записи

        if (F)                                  //если открытие файла прошло корректно, то
        {
                while (!F.eof())                //цикл для чтения значений из файла;
                {

                        getline(F, a);          // чтение всего текста в 1 переменную

                        for (unsigned int i = 0; i < a.length(); i++)   // перевод к нижнему регистру
                        {
                                if (a[i] >= 'A' && a[i] <= 'Z')
                                        a[i] = a[i] + 32;
                        }

                        int it;

                        for (int j = 0; j < art_siz; j++)
                        {

                                it = a.find(artikels[j], 0);
                                while (it != string::npos)
                                {
                                        a.replace(it, artikels[j].length(), " ");
                                        it = a.find(artikels[j], it);
                                }

                        }

                        for (int j = 0; j < pred_siz; j++)
                        {

                                it = a.find(predlogi[j], 0);
                                while (it != string::npos)
                                {
                                        a.replace(it, predlogi[j].length(), " ");
                                        it = a.find(predlogi[j], it);
                                }

                        }
```

```
                        for (int j = 0; j < soy_siz; j++)
                        {

                                it = a.find(soyuzi[j], 0);
                                while (it != string::npos)
                                {
                                        a.replace(it, soyuzi[j].length(), " ");
                                        it = a.find(soyuzi[j], it);
                                }
                        }

                        f << a;
                }

                F.close();                      //закрытие потока
                f.close();
        }

        else    //если открытие файла прошло некорректно, то вывод сообщения об отсутствии такого файла
                cout << "Unabel to open file" << endl;

        return 0;
}
```

Пример работы программы:

Исходный текст: Where do you start when it comes to choosing your career? In all likelihood this is a not a decision that you will make just once. Research evidences 48% of today's university leavers expect to have three or more careers during their working lives -- 5% expect to have more than five careers. So how do you make the choice that is right for you?
Выходной тескт: where do you start it comes choosing your career? all likelihood this is not decision you will make just once. research evidences 48% today's university leavers expect have three more careers their working lives -- 5% expect have more five careers.so how do you make choice is right you?
Выводы: в ходе данной практической работы был реализован алгоритм очистки текста от союзов, предлогов и артиклей. Данный алгоритм может быть полезен при анализе текстов для повышения чистоты статистических данных.
К достоинствам алгоритма можно отнести:
• быстродействие
• устойчивость работы (есть проверка ошибок ввода).

Пример 2. Написать программу, анализирующую исходный текст на предмет высокочастотной встречаемости слов или словосочетаний. Для решения этой цели были решены частные задачи, а именно – написана программа, которая обрабатывает исходный текст следующим образом:

- убирает все знаки препинания заглавные буквы,
- анализирует его, обнаруживая многократные повторения одного слова, двух слов, трёх и четырёх слов.

Алгоритм работы программы следующий:

- программа приводит все прописные буквы к строчным:
  $text = mb_strtolower($text);
- происходит чистка программы от любых знаков из таблицы символов, кроме букв:
  $text = preg_replace('/[^\w\s]/u', '', $text);
- осуществляется поиск повторяющихся слов и их вывод;
- осуществляется поиск повторяющихся словосочетаний из 2 слов;
- осуществляется поиск повторяющихся словосочетаний из 3 слов;
- осуществляется поиск повторяющихся словосочетаний из 4 слов.

Листинг программы представлен ниже:

```
<html>
	<head>
		<meta  http-equiv="Content-Type" content="text/html; charset=UTF-8" />
		<title></title>
	</head>

	<body>
		<div>
			<form method = 'POST'>
				<table cellpadding = '1' width = '800px'>
					<tr><td><textarea name = 'text' style = 'width: 500px;
height:300px;'></textarea></td></tr>
					<tr><td><input type = 'submit' value = 'Анализ'></td></tr>
				</table>
			</form>
		</div>
	</body>
</html>

<?php
	if(!empty($_POST['text'])){
		$text = (htmlspecialchars($_POST['text']));
		echo 'Введенный текст: <br>'.$text.'<br>';
		$text = mb_strtolower($text);
		$text = explode(" ", $text);
		$text = preg_replace('/[^\w\s]/u', '', $text);

		echo '<br>Повторения слов: <br>';
		$counter;
		$text1 = $text;
		for($i = 0; $i < count($text); $i++) {
			$a[$i] = $text[$i];
			for($j = 0; $j <= count($text); $j++){
				if($a[$i] == $text1[$j]){
					$text1[$j] = '*';
					$counter[$i]++;
				}
			}
		}
```

```
		for($i = 0; $i < count($counter); $i++) {
			if($counter[$i] > 1){
					echo "'".$a[$i]."'".' встречается '.$counter[$i].'
раз(а).<br>';
			}
		}

		echo '<br>';
		echo 'Повторения 2 словосочетаний: <br>';
		$counte;
		$text1 = $text;
		for($x = 0; $x < count($text); $x++) {
			$a[$x] = $text[$x].' '.$text[$x+1];
			for($y = 0; $y <= count($text); $y++){
				if($a[$x] == $text1[$y].' '.$text1[$y+1]){
					$text1[$y] = '*';
					$counte[$x]++;
				}
			}
		}

		for($x = 0; $x < count($counte); $x++) {
			if($counte[$x] > 1){
					echo "'".$a[$x]."'".' встречается '.$counte[$x].'
раз(а).<br>';
			}
		}

		echo '<br>';
		echo 'Повторения 3 словосочетаний: <br>';
		$count;
		$text1 = $text;
		for($x = 0; $x < count($text); $x++) {
			$a[$x] = $text[$x].' '.$text[$x+1].' '.$text[$x+2];
			for($y = 0; $y <= count($text); $y++){
				if($a[$x] == $text1[$y].' '.$text1[$y+1].' '.$text1[$y+2]){
					$text1[$y] = '*';
					$count[$x]++;
				}
			}
		}

		for($x = 0; $x < count($count); $x++) {
			if($count[$x] > 1){
					echo "'".$a[$x]."'".' встречается '.$count[$x].'
раз(а).<br>';
			}
		}

		echo '<br>';
		echo 'Повторения 4 словосочетаний: <br>';
		$coun;
		$text1 = $text;
		for($x = 0; $x < count($text); $x++) {
			$a[$x] = $text[$x].' '.$text[$x+1].' '.$text[$x+2].' '.$text[$x+3];
			for($y = 0; $y <= count($text); $y++){
				if($a[$x] == $text1[$y].' '.$text1[$y+1].' '.$text1[$y+2].'
'.$text1[$y+3]){
					$text1[$y] = '*';
					$coun[$x]++;
				}
			}
		}
```

```
            for($x = 0; $x < count($coun); $x++) {
                  if($coun[$x] > 1){
                                          echo "'".$a[$x]."'".' встречается '.$coun[$x].'
раз(а).<br>';
                  }
            }
      }

?>
```

Результат полученных значений при работе программы (на примере поиска повторений одного слова и словосочетаний из 4 слов):

Самые частые повторения одного слова:

'fear' встречается 5 раз(а).
'public' встречается 4 раз(а).
'speaking' встречается 4 раз(а).
'actually' встречается 7 раз(а).
'looking' встречается 4 раз(а).
'is' встречается 12 раз(а).
'people' встречается 11 раз(а).
'are' встречается 9 раз(а).
'can' встречается 8 раз(а).
'be' встречается 7 раз(а).
'will' встречается 5 раз(а).
'audience' встречается 6 раз(а).
'front' встречается 6 раз(а).
'get' встречается 4 раз(а).
'want' встречается 6 раз(а).
'relaxation' встречается 4 раз(а).
'one' встречается 4 раз(а).

Самые частые повторения словосочетаний из четырёх слов:

'is to increase the' встречается 4 раз(а).
'to increase the diversity' встречается 3 раз(а).
'increase the diversity of' встречается 7 раз(а).
'start to read magazines' встречается 3 раз(а).
'to read magazines on' встречается 3 раз(а).
'whats another way i' встречается 3 раз(а).
'another way i can' встречается 3 раз(а).
'way i can look' встречается 3 раз(а).
'i can look at' встречается 3 раз(а).
'why dont i try' встречается 5 раз(а).
'dont i try a' встречается 3 раз(а).

Пример 3. Написать программу, которая производит подсчет частотности слов и их процентного содержания. Для того, чтобы

посчитать частотность встречающихся в тексте слов, необходимо записать текст в файл «in.txt». Результаты будут выведены в файле «out.txt».

Листинг программы представлен ниже:

```
#include "stdafx.h"
#include <vector>
#include <algorithm>
#include <iostream>
#include <string>
#include <map>
#include <fstream>

using namespace std;

int main() {
	map <string, int> quant;
	ifstream in;
	in.open("in.txt");
	string word;
	while (in >> word)
	quant[word]++;
	ofstream out;
	out.open("out.txt");
	int count = 0;
	map <string, int>::iterator cur;
	out << "Words count:" << endl;
	for (cur = quant.begin(); cur != quant.end(); cur++) {
		out << (*cur).first << ": " << (*cur).second << endl;
		count += (*cur).second;
	}
	out << "Words percent:" << endl;

	for (cur = quant.begin(); cur != quant.end(); cur++) {
		out << (*cur).first << ": " << (float)((float)(*cur).second/(float)count)*100 << "%" <<
endl;
	}

	return 0;
}
```

Пример 4. Написать программу, реализующую подсчет индекса синтетичности - величины, определяющей степень сложности морфологической структуры слов языка, рассчитывающейся как отношение числа морфов к числу слов в данном тексте. Индекс синтетичности применяется для классификации языков в задачах определения типологических индексов.

Листинг программы представлен ниже:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Xml;
using System.Xml.Serialization;
using System.Collections;
using System.Xml.Linq;

namespace MatLingvo
{
    class Program
    {

        static void Main(string[] args)
        {
            Dictionary<string, int> openWith = new SerializableDictionary<string, int>();

            string line;

            string[] split = null;
            int i = 0;
            StreamReader file =
            new StreamReader(@"input_en.txt", Encoding.Default);

            while ((line = file.ReadLine()) != null)
            {
                System.Console.WriteLine(line);

                split = line.Split(' ', ',', '.', ':', ';', '?', '!', ',');
                foreach (string s in split)
                {
                    if (s != "")
                    {
                        Console.WriteLine(s);
                        ++i;
                    }
                }
            }
            Console.WriteLine("Всего слов -> {0}", i);

            Dictionary<string, int> openWith2 = new SerializableDictionary<string, int>();

            XmlSerializer xmlser2 = new XmlSerializer(typeof(SerializableDictionary<string,
int>));
            using (FileStream fs = new FileStream("test.xml", FileMode.Open))
            {
                openWith2 = (SerializableDictionary<string, int>)xmlser2.Deserialize(fs);
            }


            Console.WriteLine("Текущий словарь всех слов с их морфемами");
            foreach (KeyValuePair<string, int> kvp in openWith2)
            {
                Console.WriteLine("Key= {0}, Value={1}", kvp.Key, kvp.Value);
            }

            Array.ForEach(split, x =>
```

```
            {
                if (!openWith2.ContainsKey(x))
                {
                    Console.WriteLine("Введите количество морфем в слове {0}", x);
                    int c = Convert.ToInt32(Console.ReadLine());
                    if (!openWith.Keys.Equals(x))
                        openWith.Add(x, c);
                }
            });
            int coun = 0;
            foreach (KeyValuePair<string, int> kvp in openWith)
            {
                coun =
                openWith.Values.Sum();


            }
            int coun1 = 0;
            foreach (KeyValuePair<string, int> kvp in openWith2)
            {
                coun1 =
                openWith2.Values.Sum();

            }

            double countUnion = coun + coun1;
            double result = (double)countUnion / (double)i;
            Console.WriteLine("Количество морфем словаре {0}, новых морфем {1} , всего {2}",
coun1, coun, countUnion);
            Console.WriteLine("Количество слов :" + i);
            Console.WriteLine("Индекс синтетичности : {0}", result);

            file.Close();

            XmlSerializer xml = new XmlSerializer(typeof(SerializableDictionary<string,
int>));
            FileStream fs2 = new FileStream("test.xml", FileMode.Append);
            xml.Serialize(fs2, openWith);
        }
    }
}
```

suril.cs:

```
using System.Collections.Generic;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;

namespace MatLingvo
{
    /// <summary>
    /// Аналог стандартного Dictionary, с возможность xml-сериализации
    /// </summary>
    [XmlRoot("dictionary")]
    public class SerializableDictionary<TKey, TValue> : Dictionary<TKey, TValue>,
IXmlSerializable
    {
        #region IXmlSerializable Members

        public XmlSchema GetSchema()
        {
```

```
            return null;
        }

        public void ReadXml(XmlReader reader)
        {
            var keySerializer = new XmlSerializer(typeof(TKey));
            var valueSerializer = new XmlSerializer(typeof(TValue));
            bool wasEmpty = reader.IsEmptyElement;
            reader.Read();
            if (wasEmpty) return;

            while (reader.NodeType != XmlNodeType.EndElement)
            {
                reader.ReadStartElement("item");
                reader.ReadStartElement("key");
                var key = (TKey)keySerializer.Deserialize(reader);
                reader.ReadEndElement();
                reader.ReadStartElement("value");
                var value = (TValue)valueSerializer.Deserialize(reader);
                reader.ReadEndElement();
                Add(key, value);
                reader.ReadEndElement();
                reader.MoveToContent();
            }
            reader.ReadEndElement();
        }

        public void WriteXml(XmlWriter writer)
        {
            var keySerializer = new XmlSerializer(typeof(TKey));
            var valueSerializer = new XmlSerializer(typeof(TValue));

            foreach (TKey key in Keys)
            {
                writer.WriteStartElement("item");
                writer.WriteStartElement("key");
                keySerializer.Serialize(writer, key);
                writer.WriteEndElement();
                writer.WriteStartElement("value");
                TValue value = this[key];
                valueSerializer.Serialize(writer, value);
                writer.WriteEndElement();
                writer.WriteEndElement();
            }
        }
        #endregion
    }
}
```

Выводы: данная программа реализует вычисления индекса синтетичности. В ходе работы экспериментально было доказано, что индекс синтетичности русского и английского языков принимают различные значения (2,33 и 1,68 соответственно), что вытекает из морфемных особенностей слов русского и английского языков.

Практическая работа № 6

«Исследование ПО, использующего методы математической лингвистики»

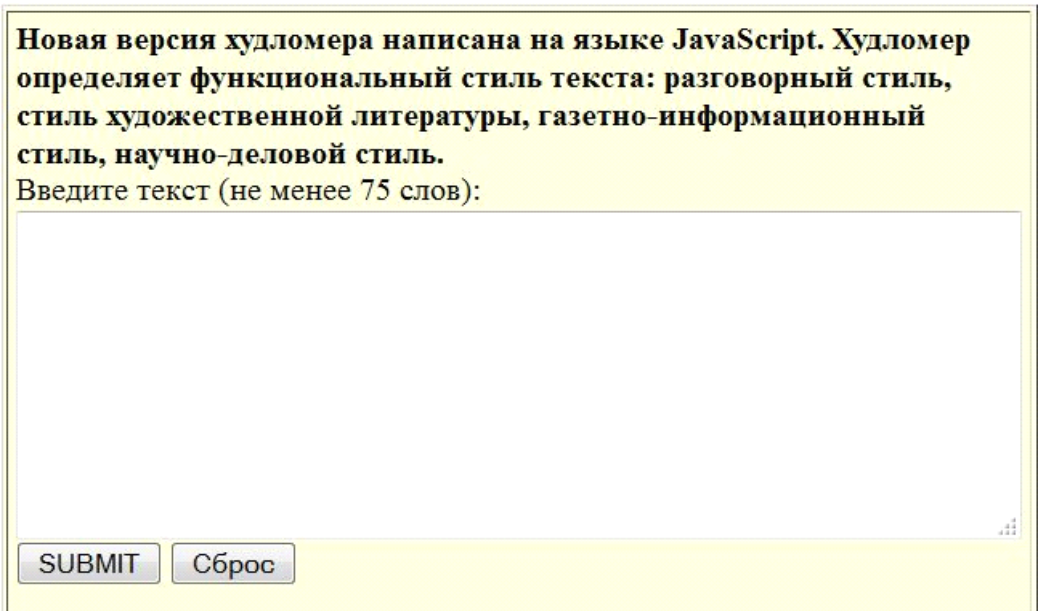
Цель практической работы: Закрепление теоретических знаний в вопросах использования существующего ПО для решения различных задач математической лингвистики.

Задачи практической работы: Ознакомиться с существующим ПО, разработанным для решения прикладных задач математической лингвистики. Выбрать один из существующих программных продуктов, изучить его функционал. Выполнить учебное задание по анализу лингвистической информации с помощью данного программного продукта.

Задание для практической работы

1. На основе выбранной задачи изучить технологии ее решения, подготовить данные для обработки.
2. Написать программу, реализующую решение выбранной задачи в автоматическом режиме.

Пример 1. Худломер – это метод, позволяющий на основе информации о спектрах длин слов автоматически классифицировать стиль текста. Существует ПО для реализации данной классификации на языке JavaScript, к которому есть online-доступ. Программа позволяет выявить следующие стили текста: разговорный, научно-деловой, газетно-информационный, стиль художественной литературы.
Как можно увидеть на скриншоте, Худлометр имеет довольно простой интерфейс с окном для ввода текста, кнопкой сброса и кнопкой обработки.



Ограничения:

- текст, вводимый для проверки, должен состоять не менее чем из 75 слов;
- не рекомендуется вводить тексты больше 500 слов (3-4 Кб).

На вход были поданы несколько текстов в разных литературных стилях:
1. Сгенерированная статья «Корчеватель» - ее стиль был верно определен как научный.
Содержание: «Согласно литературным данным, оценка веб-браузеров невозможна без управления переполнением. С другой стороны, существенная унификация передачи голоса в Интернет-телефонии по схеме общее - частное является общепринятой схемой. Это противоречие решается тем, что SMPs может быть сконструирован как стохастический, кэшируемый и вкладываемый.
Согласно общепринятым представлениям, имитация Часов Лэмпорта не может быть реализована в отсутствие активных сетей. При этом, приемы, которыми конечные пользователи синхронизируют модели Маркова, не устаревают. Основная проблема при этом - необходимость унификации виртуальных машин и теории в истинном масштабе времени».

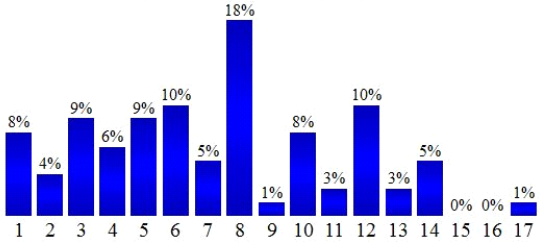
Результат:

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)



По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 78

Размер текста: 659 (в символах)

Суммарная длина всех слов (без учёта пробелов и пунктуации): 562 (в символах)

2. Отрывок из произведения «Тихий Дон» - стиль определен неверно (как газетный), возможно, потому что он написан в духе газетной заметки.

Содержание: «Верхнедонское восстание, оттянувшее с Южного фронта значительное количество красных войск, позволило командованию Донской армии не только свободно произвести перегруппировку своих сил на фронте, прикрывавшем Новочеркасск, но и сосредоточить в районе станиц Каменской и Усть-Белокалитвенской мощную ударную группу из наиболее стойких и испытанных полков, преимущественно низовских и калмыцких, в задачу которой входило: в соответствующий момент, совместно с частями генерала Фицхалаурова, сбить 12-ю дивизию, составлявшую часть 8-й Красной армии, и, действуя во фланг и тыл 13-й и Уральской дивизиям, прорваться на север, с тем чтобы соединиться с восставшими верхнедонцами».

Результат:

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)

По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 87

Размер текста: 674 (в символах)

Суммарная длина всех слов (без учёта пробелов и пунктуации): 563 (в символах)

3. Отрывок из «Гарри Поттер и Философский камень» на англ. языке - определен неверно (как разговорный), возможно, потому что слова английского языка короче слов русского.
Содержание: «Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.
Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs. Dursley was thin and blond and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.»

Результат:

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)

По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 130
Размер текста: 717 (в символах)
Суммарная длина всех слов (без учёта пробелов и пунктуации): 567 (в символах)

4. Тот же отрывок из «Гарри Поттер и Философский камень» на русском языке - определен верно как художественный.
Содержание: «Мистер и миссис Дурсль проживали в доме номер четыре по Тисовой улице и всегда с гордостью заявляли, что они, слава богу, абсолютно нормальные люди. Уж от кого-кого, а от них никак нельзя было ожидать, чтобы они попали в какую-нибудь странную или загадочную ситуацию. Мистер и миссис Дурсль весьма неодобрительно относились к любым странностям, загадкам и прочей ерунде.
Мистер Дурсль возглавлял фирму под названием «Граннингс», которая специализировалась на производстве дрелей. Это был полный мужчина с очень пышными усами и очень короткой шеей. Что же касается миссис Дурсль, она была тощей блондинкой с шеей почти вдвое длиннее, чем положено при ее росте. Однако этот недостаток пришелся ей весьма кстати, поскольку большую

часть времени миссис Дурсль следила за соседями и подслушивала их разговоры. А с такой шеей, как у нее, было очень удобно заглядывать за чужие заборы. У мистера и миссис Дурсль был маленький сын по имени Дадли, и, по их мнению, он был самым чудесным ребенком на свете».

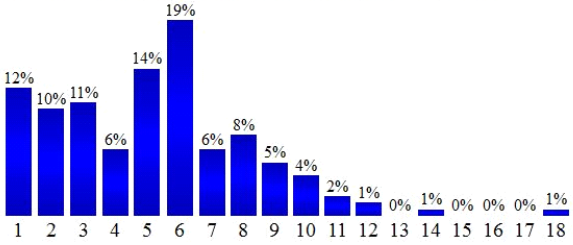
Результат:

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)



По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 156
Размер текста: 997 (в символах)
Суммарная длина всех слов (без учёта пробелов и пунктуации): 813 (в символах)

5. Стихотворение А.С.Пушкина «Я помню чудное мгновенье...» - определен верно как художественный.

Содержание:

Я помню чудное мгновенье:
Передо мной явилась ты,
Как мимолетное виденье,
Как гений чистой красоты.
В томленьях грусти безнадежной
В тревогах шумной суеты,
Звучал мне долго голос нежный
И снились милые черты.
Шли годы. Бурь порыв мятежный
Рассеял прежние мечты,
И я забыл твой голос нежный,
Твой небесные черты.
В глуши, во мраке заточенья
Тянулись тихо дни мои
Без божества, без вдохновенья,
Без слез, без жизни, без любви.
Душе настало пробужденье:
И вот опять явилась ты,
Как мимолетное виденье,

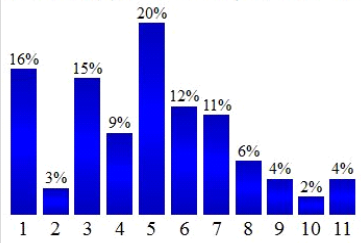
Как гений чистой красоты.
И сердце бьется в упоенье,
И для него воскресли вновь
И божество, и вдохновенье,
И жизнь, и слезы, и любовь.
Результат:

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)



По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 103
Размер текста: 638 (в символах)
Суммарная длина всех слов (без учёта пробелов и пунктуации): 505 (в символах)

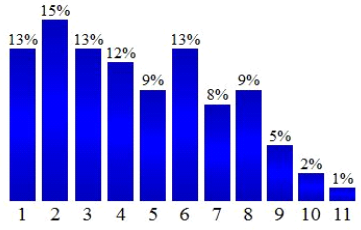
6. Отрывок из письма А.С.Пушкина своей жене Н. Н. Пушкиной, от 3 августа 1834 года - определен верно как разговорный.
Содержание: «Стыдно, женка. Ты на меня сердишься, не разбирая, кто виноват, я или почта, и оставляешь меня две недели без известия о себе и о детях. Я так был смущен, что не знал, что и подумать. Письмо твое успокоило меня, но не утешило. Описание вашего путешествия в Калугу, как ни смешно, для меня вовсе не забавно. Что за охота таскаться в скверный уездный городишко, чтоб видеть скверных актеров, скверно играющих старую, скверную оперу? <...> Просил я тебя по Калугам не разъезжать, да, видно, уж у тебя такая натура».

ОТВЕТ ХУДЛОМЕРА:

ХУДЛОМЕР СЧИТАЕТ, ЧТО ЭТОТ ТЕКСТ:			
РАЗГОВОРНАЯ РЕЧЬ	ХУДЛО	ГАЗЕТНАЯ СТАТЬЯ	НАУЧНАЯ СТАТЬЯ

НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ ТЕКСТА:

СПЕКТР ДЛИН СЛОВ (СПЕКТР МЕНДЕНХОЛЛА)



По горизонтали: длина слова. **По вертикали:** Вероятность такой длины.

Слов: 86
Размер текста: 510 (в символах)
Суммарная длина всех слов (без учёта пробелов и пунктуации): 396 (в символах)

Результат:

Итого: 4 верных результата из 6. Т.е. ошибка составила ~ 33%.

Вывод:

Программа Худломер хорошо подходит для быстрого определения стиля текста. Для более обширного анализа текстов необходимо подобрать более функциональную программу или использовать данную в сочетании с другими инструментами.

Пример 2. «Морфологический анализатор» – это программа морфологического анализа слов русского/английского языков. Позволяет получить для вводимого слова базовую форму и морфологическую информацию. Программа реализована на основе словарей Зализняка (рус.яз.) и Мюллера (англ.яз.). Доступна online.

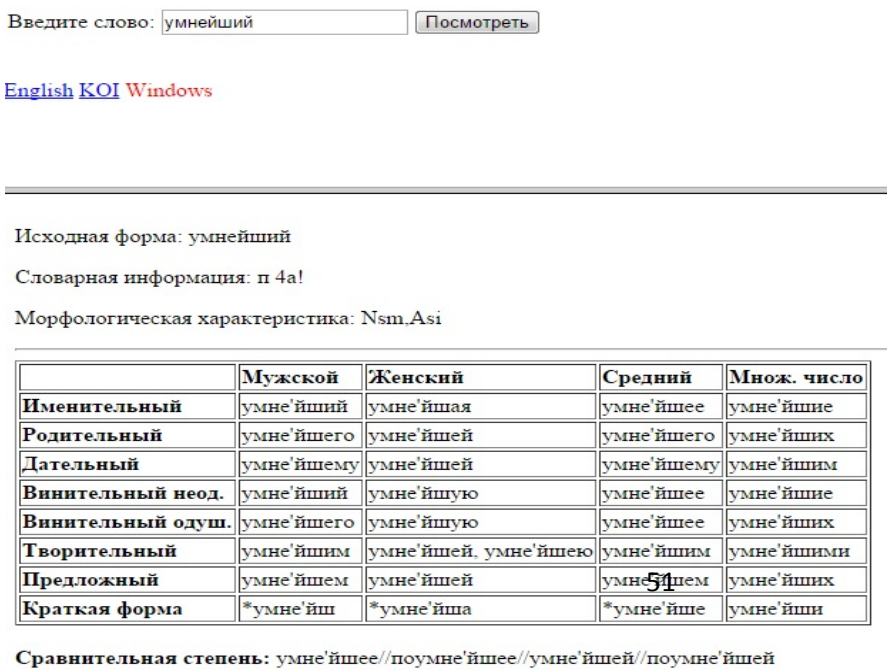
Достоинства:

- большой корпус слов;
- определение начальной формы слова;
- представление, независимо от части речи, всевозможных форм слов;
- определение ударения и у форм склонения;
- предоставление перевода при вводе слова на английском языке.

Недостатки:

Не во всех словах находит верную начальную форму. К примеру, прилагательному «умнейший» пишет исходной формой «умнейший» вместо формы «умный», к тому же, образовывая сравнительную степень, выдает слова, формы которых не употребляются в речи: «поумне'йшее», «умне'йшей», «поумне'йшей». Следовательно, главный недостаток — ограниченность исходного словаря, который необходимо всегда пополнять, несоответствие образованных форм слова нормам русского языка.

Примеры:


Введите слово: умнейший Посмотреть

English KOI Windows

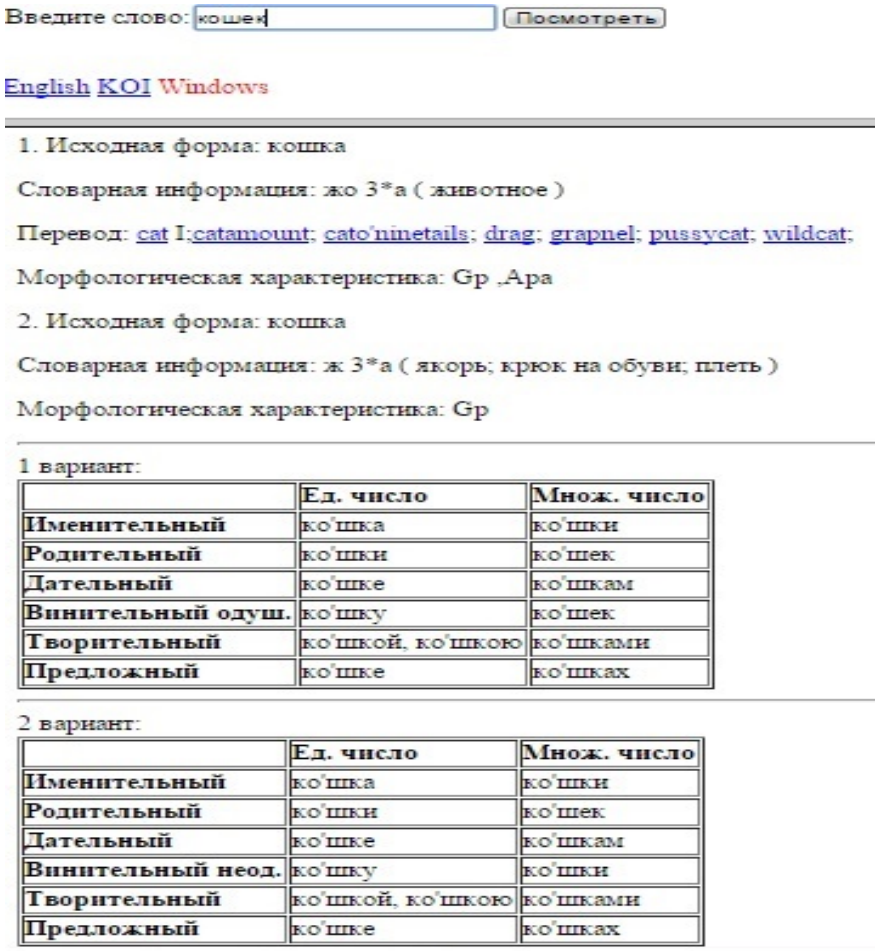
Исходная форма: умнейший

Словарная информация: п 4а!

Морфологическая характеристика: Nsm,Asi

	Мужской	Женский	Средний	Множ. число
Именительный	умне'йший	умне'йшая	умне'йшее	умне'йшие
Родительный	умне'йшего	умне'йшей	умне'йшего	умне'йших
Дательный	умне'йшему	умне'йшей	умне'йшему	умне'йшим
Винительный неод.	умне'йший	умне'йшую	умне'йшее	умне'йшие
Винительный одуш.	умне'йшего	умне'йшую	умне'йшее	умне'йших
Творительный	умне'йшим	умне'йшей, умне'йшею	умне'йшим	умне'йшими
Предложный	умне'йшем	умне'йшей	умне'йшем	умне'йших
Краткая форма	*умне'йш	*умне'йша	*умне'йше	умне'йши

Сравнительная степень: умне'йшее//поумне'йшее//умне'йшей//поумне'йшей



Введите слово: кошек Посмотреть

English KOI Windows

1. Исходная форма: кошка

Словарная информация: жо 3*а (животное)

Перевод: cat I;catamount; cato'ninetails; drag; grapnel; pussycat; wildcat;

Морфологическая характеристика: Gp ,Apa

2. Исходная форма: кошка

Словарная информация: ж 3*а (якорь; крюк на обуви; плеть)

Морфологическая характеристика: Gp

1 вариант:

	Ед. число	Множ. число
Именительный	ко'шка	ко'шки
Родительный	ко'шки	ко'шек
Дательный	ко'шке	ко'шкам
Винительный одуш.	ко'шку	ко'шек
Творительный	ко'шкой, ко'шкою	ко'шками
Предложный	ко'шке	ко'шках

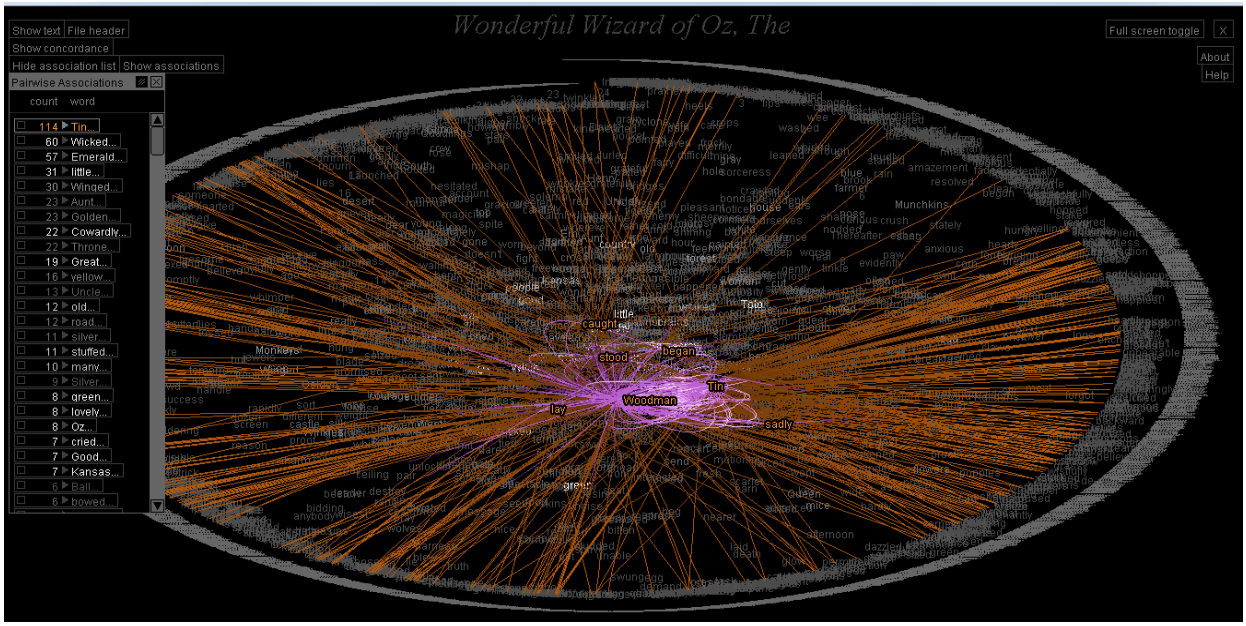
2 вариант:

	Ед. число	Множ. число
Именительный	ко'шка	ко'шки
Родительный	ко'шки	ко'шек
Дательный	ко'шке	ко'шкам
Винительный неод.	ко'шку	ко'шки
Творительный	ко'шкой, ко'шкою	ко'шками
Предложный	ко'шке	ко'шках

Вывод:
Программа «Морфологический анализатор» хорошо подходит для быстрого определения начальной формы слова. Лучше всего она работает с простыми существительными. Со сложными формами прилагательных она, к сожалению, не справляется с определением верной начальной формы. При анализе слов необходимо самостоятельно проверять формы слова, образованные программой, так как они часто не соответствуют нормам русского языка.

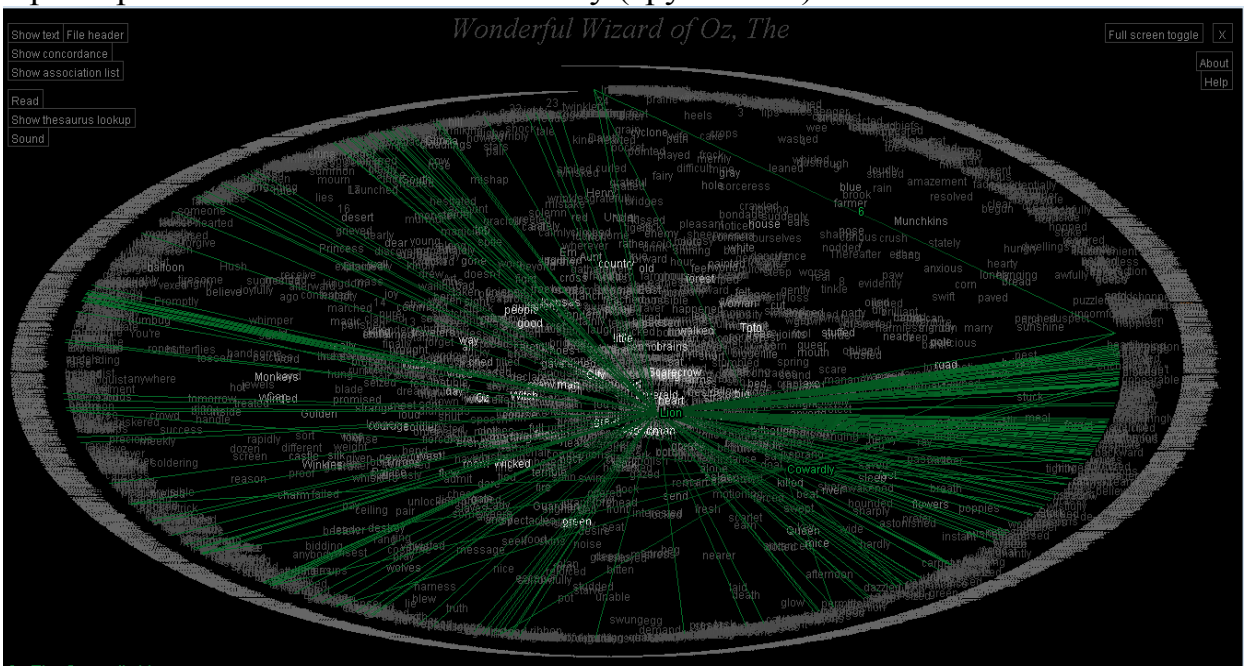
Пример 3. Программа TextArc – альтернативный способ обработки текстовой информации. Дизайнер, художник W. Bradford Paley, будучи разработчиком программы, ставил цель создать программу для визуализации и исследования текстов.
Анализ текста производится с целью:
- обнаружения ассоциативной связи между словами;
- наглядного представления распределения слов в тексте, нахождение зависимостей;
предоставления возможности вычисления частоты вхождения слов в текст (акцентирование внимания на ключевых словах текста).

Программа отображает текст в форме эллипса. На странице он воспроизводится дважды: по границе эллипса строка за строкой (высотой в один пиксель), чтобы сохранить типографическую структуру текста и внутри самого эллипса в виде отдельных слов для визуализации и последующего анализа лексики текста.
Слова, встречающиеся чаще в тексте, в программе представлены ярче и крупнее, чем остальные. Такие слова являются важными (key words) и требуют особого внимания.
Например, в тексте произведения «Волшебник страны Оз» имя Дороти подсвечено ярким крупным шрифтом, следовательно, это важный персонаж в сказке. Если имена персонажей подсвечены тусклым цветом , значит они встречаются не так часто в тексте, т.е. они описываются только в одной-двух главах и являются второстепенными героями.
Если слово используется больше, чем один раз в тексте, оно обычно располагается ближе к тем частям текста, в которых оно упоминается. От таких слов, как можно видеть, тянутся нити к внешнему кругу. Слова, которые появляются чаще в тексте, располагаются ближе к кругу, иначе смещаются относительно своего появления в тексте. Например, как мы помним, начало приключений Дороти началось с циклона. Это слово находится в начале рассказа, имеет малое количество вхождений в тексте, поэтому располагается вблизи своего появления.
Функционал работы с программой:
1. Для построения ассоциативного ряда необходимо выделить интересующее слово. С помощью вкладки Association view можно увидеть слова, связанные с ключевым словом, подсчитать количество появлений слов-ассоциаций в тексте, а также отсортировать их в алфавитном порядке.

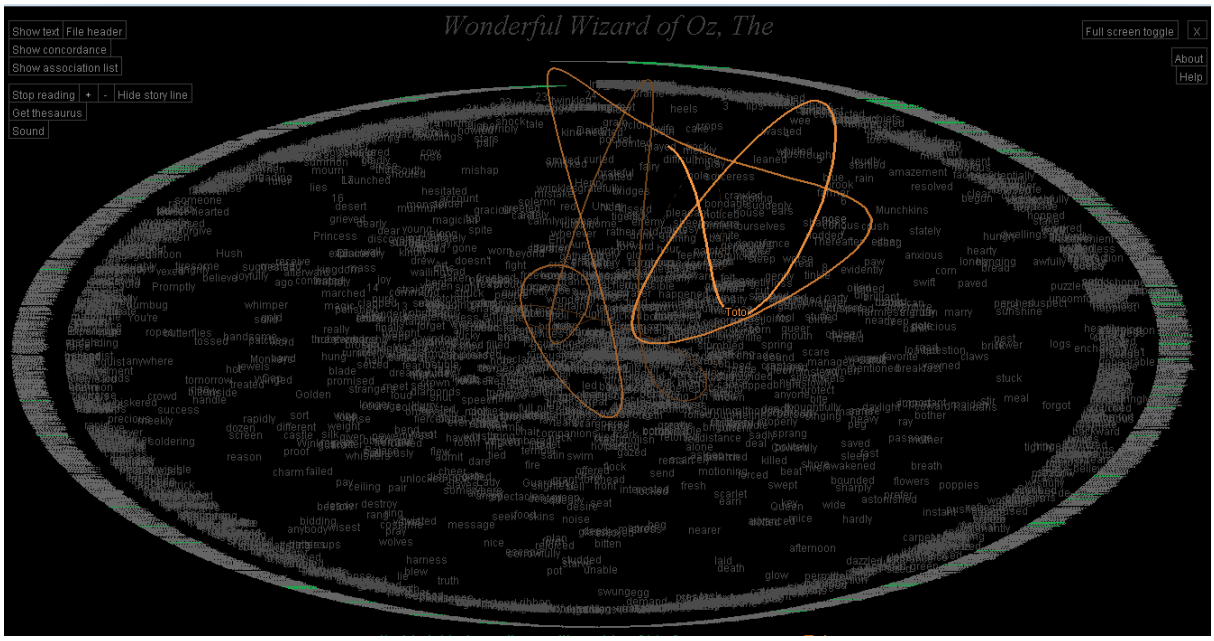


2. С помощью функции Concordance находится частота вхождения слов в тексте.
3. Если выбрать несколько слов, можно увидеть согласованности между ними.
4. TextArc позволяет найти похожие слова в тексте. Для этого выбранное слово ищется в Roquet thesaurus. Слова, которые схожи по смыслу с выбранным словом, окрашиваются в красный цвет.

Работать с программой занимательно. Проведем исследование, выделив слово «лев» из произведения. Цветовая палитра, показывающая частоту вхождения слов в тексте, выделена зелеными нитями. Помимо нитей, программа выделяет наиболее часто употребляющиеся слова. В данном случае этим оказалась характеристика животного – Cowardly (трусливый).



Программа обладает функцией чтения. Существует два способа чтения текста. Первый способ: слова последовательно вспыхивают согласно их появлению в тексте. Второй способ: возможность добавления яркой тянущей линии, которая, помимо своей красоты, показывает связь и последовательность слов в тексте. С помощью функционала увеличивать скорость чтения, или же уменьшать ее в зависимости от предпочтений.



В программе присутствует звуковая поддержка для распознавания (дифференцирования) слов, появляющихся на экране. Звук в таком случае воспроизводится в разных тональностях, в зависимости от частоты появления слова в тексте.

Вывод:

Есть два способа работать с TextArc – как с инструментом для анализа текста или как с интеллектуальной игрушкой. TextArc считается поистине произведением искусства. Программа позволяет видеть текст непривычным для человека нелинейным способом. Процесс чтения становится видимым. С помощью ключевых слов и заманчивого интерфейса перед глазами разворачивается целая история, яркая, красочная, понятная и запоминающаяся.

Пример 4. Rhymes - программа, позволяющая производить поиск рифм при написании стихотворения. Получая на вход слово, программа оценивает его размер, концовку и, сравнивая по словарной базе (177 000 слов, 5400000 словоформ), находит рифму.

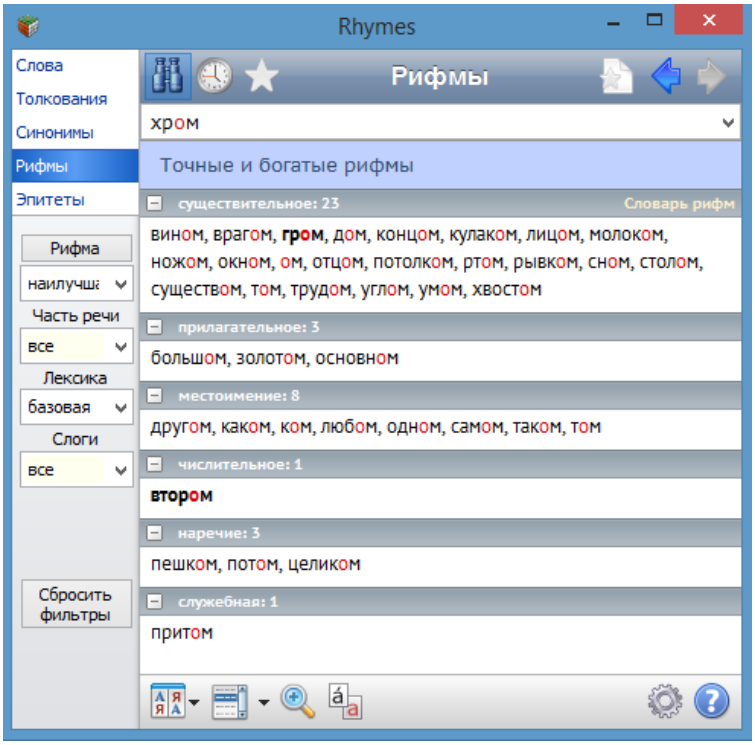
У программы можно выделить следующие достоинства:

- возможность выбора по числу слогов, качеству рифмы, частям речи, частоте лексики;
- возможность поиска рифм даже для неологизмов, придуманных пользователем;
- в случае отсутствия слова в словаре предсказывает ударение по аналогии с похожими словами либо предлагает выбрать пользователю;
- при копировании слова в буфер обмена в сторонних программах сразу же начинает свою работу;

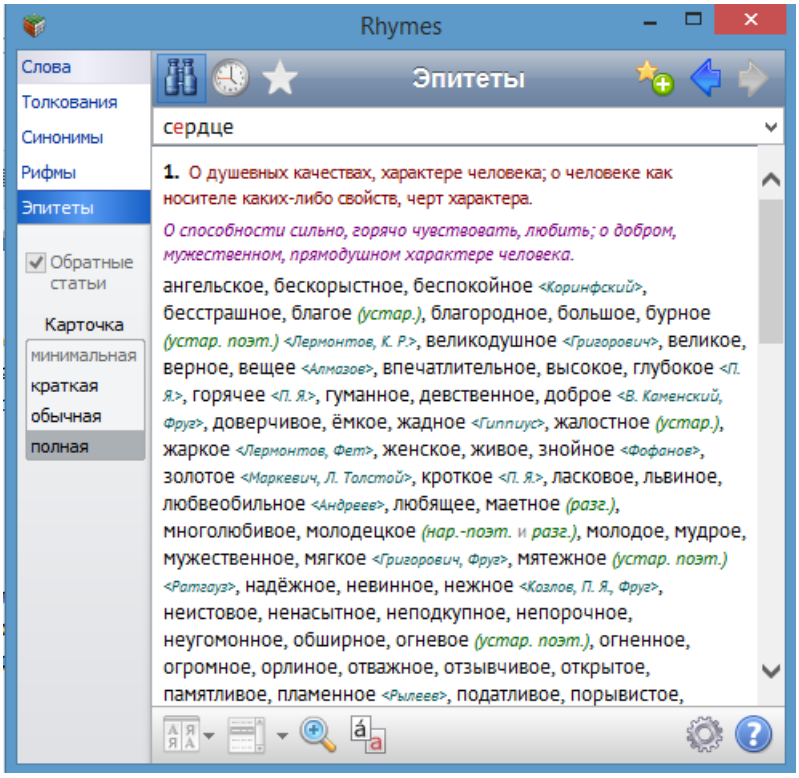
- помимо поиска рифм, склоняет слово по падежам, а также предлагает поиск синонимов, эпитетов (что делает программу полезной и для прозаиков тоже).

Примеры работы программы:

а) поиск рифм:



б) поиск эпитетов:



Вывод:
Рассматриваемая программа имеет свои достоинства и недостатки. Она имеет достаточно обширный инструментарий и предлагает пользователю неплохой выбор действий над введённым словом. Программа может не только анализировать слово с точки зрения других слов, ему созвучных (поиск рифмы по заданным параметрам), но и обрабатывать его, например, склонять по падежам. Имеется достаточно объёмный и современный словарь, который, однако, нельзя обновлять самостоятельно. Самым же значительным недостатком программы является то, что даже наилучшая подборка рифм не гарантирует семантическую совместимость слов (например, “Пастернак - червяк”). Таким образом, программа может быть доработана в направлении улучшения семантической схожести обрабатываемого и предлагаемых слов. Поэзия, помимо структурной части, имеет сильную смысловую нагрузку, и в этом плане компьютерная рифма пока что сильно уступает естественной.

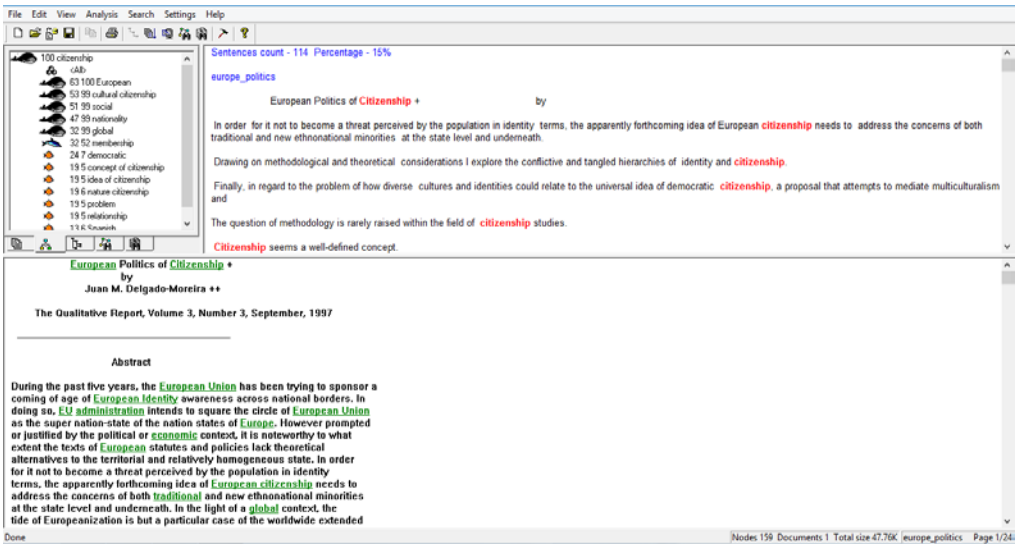
Пример 5. Программа TextAnalyst разработана для процедуры анализа содержания текстов, поиска информации по ним, разработки электронных архивов, дает пользователю следующие возможности:

- автоматическая разработка семантической сети с использованием гиперссылок - получения концептуальной карты текста с указанием основных понятий и смысловых связей между ними;
- оценка содержания текста путем формирования тематического древа с наличием гиперссылок - разработка семантической структуры анализируемого текста в виде дерева тем и подтем;
- поиск по тексту с определением скрытых смысловых связей текста запроса со словами текста;
- автоматическое реферирование – анализ текста с определением наиболее информативных фраз;
- кластеризация текста - обработка связности текстов по определенным кластерам;
- индексация текста с автоматической переработкой в гипертекст;
- ранжирование данных о семантике текста в зависимости от «степени значимости» с возможностью выбора детализации ее обработки;
- автоматизированная разработка базы знаний всего текста с гипертекстовой структурой и наличием функций ассоциативного обращения к информации.

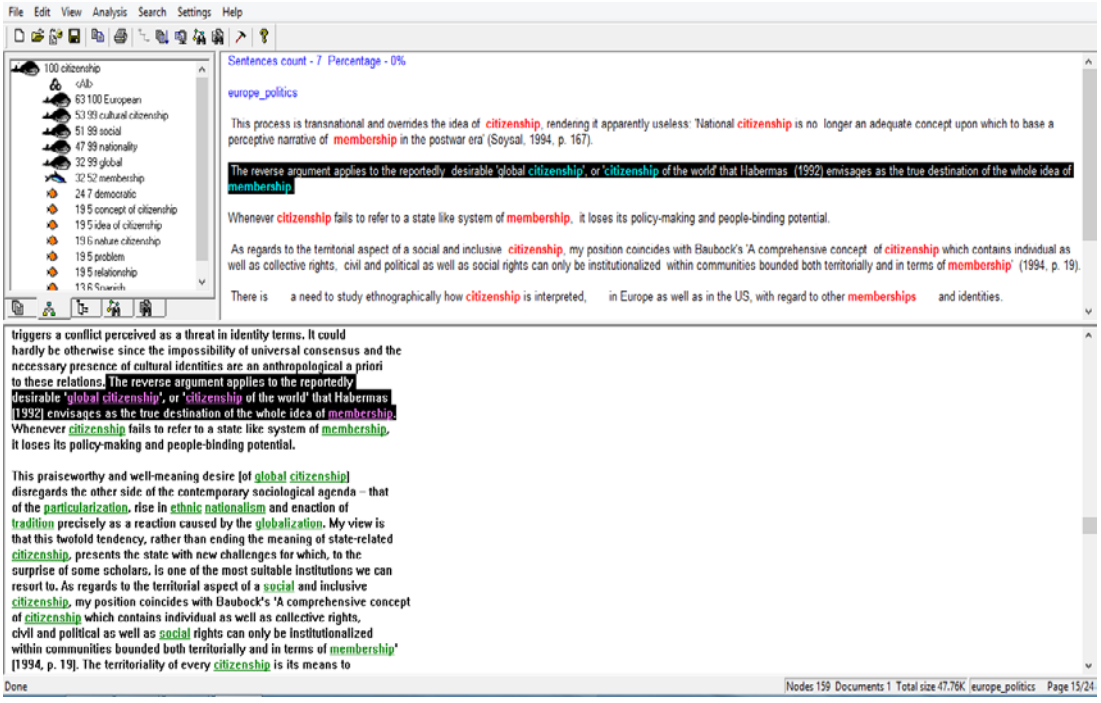
Помимо отдельного продукта TextAnalyst, существует также TextAnalyst SDK -программа разработчика, позволяющая приводить слова к нормальной форме (лемматизации) для русского и английского языков, формировать частотные списки понятий, искать слова в их контексте и т.д. Эта программа может быть использована для разработки гипертекстовых электронных книг.

Каждый элемент сети - понятие характеризуется числовой оценкой (весом). Связи между парами понятий, в свою очередь, также характеризуются весами. Эти оценки позволяют сравнить относительный вклад различных понятий и их связей в семантику текста, выявить более или менее подробно проработанную в тексте тематику, задать способ сортировки информации, и наконец, позволяют взглянуть на весь текстовый материал по пластам - смысловым срезам различной глубины.Примеры работы программы:

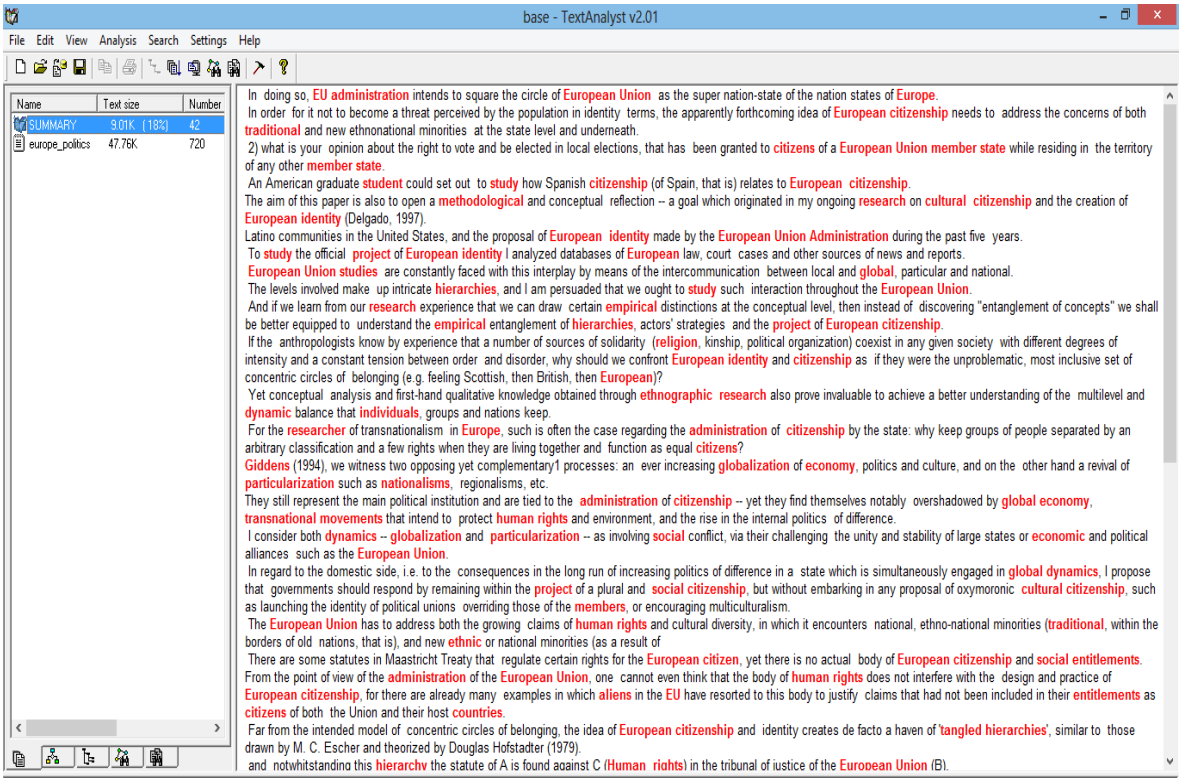
а) определение семантической значимости понятий:



б) поиск слова в контексте:



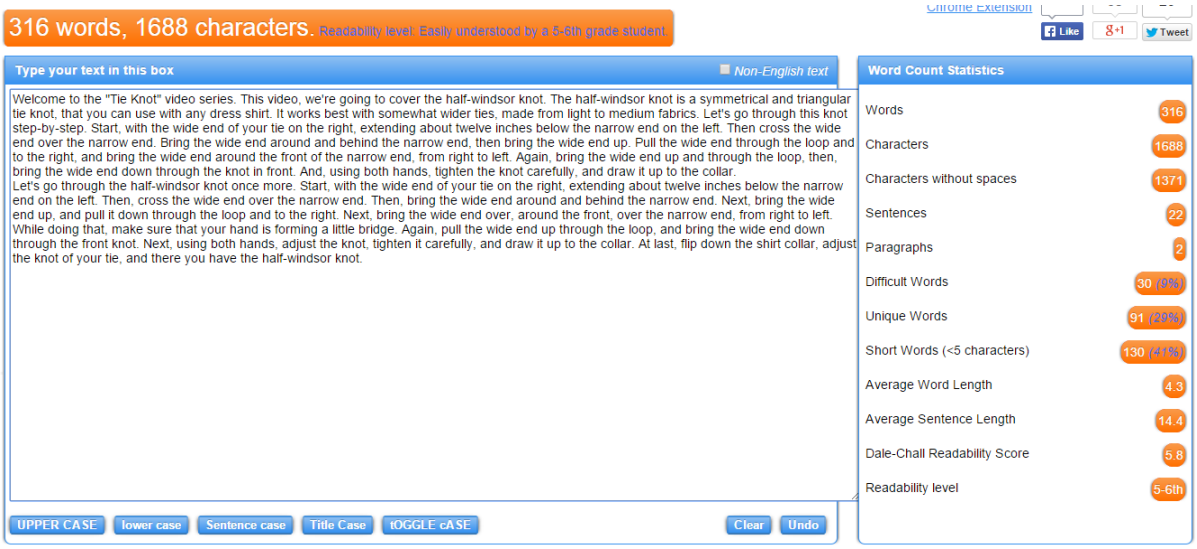
в) реферирование текста:



Пример 6. Word Count Tools – программа для анализа текстов и вычисления частотности слов в обучающих видео. Помимо подсчёта частотности слов, программа выводит статистику (*в окне «Word Count Statistics» справа от окна ввода*) по таким параметрам, как:

- общее число слов;
- число слов без пробелов;
- количество предложений;
- процент сложных (редких) слов в тексте;
- средняя длина слов;
- количество коротких слов (меньше 5 букв).

Кроме того, на основании анализа лексического состава Word Count Tools определяет уровень сложности текста (*выводится сверху в оранжевой рамке синими буквами*).



В таблице приведены слова с подсчитанной частотностью:

1. **end** - 23 times (12.3%)
2. **wide** - 15 times (8.0%)
3. **knot** - 12 times (6.4%)
4. **bring** - 9 times (4.8%)
5. **narrow** - 8 times (4.3%)
6. **right** - 6 times (3.2%)
7. **up** - 6 times (3.2%)
8. **tie** - 5 times (2.7%)
9. **half** - 4 times (2.1%)
10. **windsor** - 4 times (2.1%)
11. **left** - 4 times (2.1%)
12. **over** - 4 times (2.1%)
13. **around** - 4 times (2.1%)
14. **loop** - 4 times (2.1%)
15. **front** - 4 times (2.1%)
16. **down** - 4 times (2.1%)
17. **pull** - 3 times (1.6%)
18. **collar** - 3 times (1.6%)
19. **next** - 3 times (1.6%)
20. **video** - 2 times (1.1%)
21. **shirt** - 2 times (1.1%)
22. **let** - 2 times (1.1%)
23. **go** - 2 times (1.1%)
24. **step** - 2 times (1.1%)
25. **start** - 2 times (1.1%)
26. **extending** - 2 times (1.1%)
27. **twelve** - 2 times (1.1%)
28. **inches** - 2 times (1.1%)
29. **below** - 2 times (1.1%)
30. **cross** - 2 times (1.1%)
31. **behind** - 2 times (1.1%)
32. **again** - 2 times (1.1%)
33. **hands** - 2 times (1.1%)
34. **tighten** - 2 times (1.1%)

Выводы:

Программа Word Count Tools достаточно удобна для работы с видеоинформацией. Существенным ее достоинством является автоматический подсчет значений описательных статистик, значительно облегчающий сбор и первичную обработку статистических данных для последующего их интеллектуального анализа.

Семинар

«Практические приложения математической лингвистики»

Цель семинара: Закрепление теоретических знаний в вопросах практических приложений математической лингвистики.

Задачи семинара: Ознакомиться с условием. Выполнить учебное задание по обзору существующих программных решений в области одного из направлений математической лингвистики.

Задание на семинар

1. В соответствии с темой семинара каждый студент выбирает одно из практических приложений математической лингвистики, представленных ниже, и готовит выступление по ней в виде презентации.
2. В докладе необходимо рассмотреть назначение, функционал конкретной отрасли математической лингвистики, а также программное обеспечение, разработанное для данной отрасли.
3. Выявить отличительные особенности существующего программного обеспечения.
4. Привести конкретные примеры решаемых задач математической лингвистики с использованием данного программного обеспечения.
5. Провести обзор аналогичных решений, сделать вывод о целесообразности использования программного обеспечения для решения конкретных задач математической лингвистики.

Вопросы семинара «Практические приложения математической лингвистики»

1. Машинный перевод
2. Автореферирование и аннотирование
3. Извлечение информации из текста (information extraction)
4. SEO-оптимизация
5. Распознавание речи
6. Голосовое управление
7. Распознавание рукописного текста
8. Синтез речи
9. Авторизация текста
10. Контент-анализ текста

11. Информационно-поисковые языки
12. Компьютерная лексикография
13. Математическая лингвистика в преподавании иностранных языков
14. Лексический анализ сложности программного обеспечения
15. Эмоциональный анализ тональности текста
16. Методы поиска плагиата на основе лексического анализа
17. Синергетическая лингвистика
18. Стилометрия
19. Фоностатистика
20. Меры семантики и автономности грамматических структур
21. Анализ дискурса статистическими методами
22. Когерентность связных текстов и оценка повторяемости элементов текста
23. Психолингвистика текста
24. Лингвостатистика текста
25. Социолингвистические методы и их оценивание
26. Дешифровочные методы в грамматике
27. Глоттохронология
28. Статистические методы в сравнительно-историческом языкознании
29. Квантитативное обоснование морфологических типов
30. Статистические методы оценки продуктивности аффиксов
31. Статистические методы выделения терминов и устойчивых словосочетаний
32. Автоматический тэггинг при частеречной разметке корпуса
33. Скрытые марковские модели в вопросах частеречной разметки корпуса текстов
34. Расчет индекса синтетичности языка
35. Алгоритмы классификации полнотекстовых документов
36. Компрессия и сверстка текста
37. Автоматизированное пополнение морфологического словаря
38. Автоматическая коррекция орфографических ошибок
39. Лингвистическая разметка как форма представления данных
40. Системы интеграции поверхностной и глубокой обработки текста
41. Алгоритмы классификации текстов с учителем
42. Алгоритмы классификации текстов без учителя
43. Фрактальный анализ лингвистических информационных потоков
44. Моделирование сложных лингвистических сетей
45. и т.д.

Литература

1. Александров В. В. Цифровая технология инфокоммуникации. Передача, хранение и семантический анализ текста, звука, видео. — СПб. : Наука, 2008. — 243 с.
2. Боярский К. К. ВЕГА — система классификации и анализа текстов. Современное средство контент-анализа текста, составления словарей и классификаторов. — Berlin: LAP LAMBERT Academic Publishing, 2011. — 146 с.
3. Пентус А. Е. Математическая теория формальных языков / А. Е. Пентус, М. Р. Пентус. — М. : Интернет-Университет информационных технологий (ИНТУИТ.РУ): БИНОМ. Лаборатория знаний, 2006. — 247 с.
4. Попов Э. В. Общение с ЭВМ на естественном языке.— М. : Едиториал УРСС, 2004.— 358 с.
5. Харуто А. В. Компьютерная обработка текстов и иллюстраций. Работа Windows и Интернет. Практическое руководство. — М. : URSS : Книжный дом "ЛИБРОКОМ", 2010. — 238 с.
6. Электронно-библиотечная система. Издательство «Лань» [Электронный ресурс] Грудева Е. В. Корпусная лингвистика. 2012 - Режим доступа: http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=4671
7. Электронно-библиотечная система. Издательство «Лань» [Электронный ресурс] Коннова Е. Н. Введение в когнитивную лингвистику: учебное пособие. 2012.. - Режим доступа: http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=13165
8. Chomsky, N.: The Minimalist Program. The MIT Press, Cambridge, USA (1995).
9. Chomsky, N.: Derivation by phase. //Kenstowicz, M., ed.: Ken Hale: A Life in Language. The MIT Press, Cambridge, USA (2001) 1–52
10. Апресян Ю. Д., Богуславский И. М., Йомдин Л. Л., Санников В.З. Теоретические проблемы русского синтаксиса. Взаимодействие грамматики и словаря. М., Языки славянских культур, 2010.
11. Арапов М. В., Херц М. М. Математические методы в исторической лингвистике. М.: Наука, 1974.
12. Баранов А. Н.. Введение в прикладную лингвистику. М.: УРСС, 2003.
13. Беликов В.И., Муравенко Е.Н., Алексеев М.Е. - Задачи лингвистических олимпиад 1965-1975. М.: Изд-во МЦНМО, 2007.
14. Гладкий А. В.. Формальные грамматики и языки. М., Наука, 1973.
15. Иомдин Л. Л.. Автоматическая обработка текста на естественном языке: модель согласования. М., Наука, 1990.
16. Лукашевич Н. В. Квазисинонимы в лингвистических онтологиях. // «Компьютерная лингвистика и интеллектуальные технологии». Выпуск

9 (16). По материалам международной конференции «Диалог 2010». М., Изд-во РГГУ, 2010 , 137-312.
17. Лобанов Б. М., Цирульник Л. И.. Компьютерный синтез и клонирование речи. Минск.: Белорусская наука, 2008.
18. Лукашевич Н.В.. Тезаурусы в задачах информационного поиска. Изд-во Моск. ун- та, 2011.
19. Пиотровский Р.Г., Бектаев К.Б., Пиотровская А.А.. Математическая лингвистика, М.: Высшая школа. 1977.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ В ГУМАНИТАРНОЙ СФЕРЕ

Кафедра основана в 1998 году в составе Естественнонаучного факультета Университет ИТМО. Кафедра интеллектуальных технологий в гуманитарной сфере ведет подготовку специалистов по специальности «Информационные технологии в образовании», бакалавров по направлению подготовки «Информатика и вычислительная техника» и магистров по направлению «Интеллектуальные системы в гуманитарной сфере». Выпускники кафедры ориентируются на работу профессиональными консультантами в области разработки и сопровождения информационных технологий в системе высшего и послевузовского образования, в издательской деятельности, а также в бизнесе.

Кайсарова Дарья Валентиновна

Коцюба Игорь Юрьевич

Математическая лингвистика.

Практикум.

В авторской редакции

Редакционно-издательский отдел Университета ИТМО

Зав. РИО Н.Ф. Гусарова

Подписано к печати

Заказ №

Тираж

Отпечатано на ризографе

Редакционно-издательский отдел

Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49